

# Secure Indoor Navigation and Operation of Mobile Robots

Dissertation  
zur  
Erlangung des akademischen Grades  
Doktor-Ingenieur (Dr.-Ing.)  
der Fakultät für Informatik und Elektrotechnik  
der Universität Rostock



Mazen Ghandour, born on 10th, May 1986 in Latakia City, Syria  
Rostock, Germany, 2016

## **Gutachter**

### 1- Gutachter

Prof. Dr.-Ing. habil. Kerstin Thurow.

Institute for Automation, Universität Rostock, Germany.

### 2- Gutachter

Prof. Dr.-Ing. Norbert Stoll.

Institute for Automation, Universität Rostock, Germany.

### 3- Gutachter

Prof. Dr. Yue Yang.

School of Traffic & Transportation Engineering, Central South University, China

Datum der Einreichung: 01.December.2016.

Datum der Verteidigung: 28.April.2017.

## Acknowledgment

Implementing this doctoral dissertation is not an individual work, rather it is the outcome of the efforts and contributions of several persons, whom I would like to express my gratitude to them.

First of all, I express my deep sense of gratitude to my supervisors Prof. Dr.-Ing. habil. Kerstin Thurow, and Prof. Dr.-Ing. Norbert Stoll. I thank them for their priceless supervisory for the whole period of my research. I was always eager to listen from them their recommendations and guidance, besides to their invaluable scientific advisory they provided to me during my research. I got from them a high experience related to scientific research methods. Moreover, I highly grateful for their social concerns about my stay in Germany, besides to their priceless efforts to fund my research period. Without their concerns, I wouldn't complete this dissertation.

Moreover, I would like to thank Prof. Dr.-Ing. Hui Liu for his scientific support he provided to me during my research. I was so grateful to know him as a friend and as a group leader who never hesitated to discuss with me scientific and practical solutions for my research in robotics.

Furthermore, I highly appreciate the unlimited support I got from my colleagues in the center for life science automation (Celisca). I am so blessed to meet experienced and specialized people, who were always open and happy to provide me with solutions and advises. Furthermore, I thank them for their help they offered to me during my stay in Germany. I mention here Dr.-Ing. Steffen Junginger, Prof. Dr.-Ing. Mohit Kumar, Dr.-Ing. Thomas Roddelkopf, Dr.-Ing. Sebastian Neubert, Dr.-Ing. habil. Heidi Fleischer, Mr. Xiangyu Gu, Mr. André Geißler, Mr. Volker Gatz, Mr. Mohamman Ruzaij, Mr. Ali Abdulla, Mr. Heiko Engelhardt, Ms. Anne Kadow, and Mr. Lars Woinar.

I would like to express my gratitude to the German society, for their humaneness, kindness and openness. I wish all the best for this amazing country and astonishing people.

Moreover, I thank all of my friends I met in Germany, for their help and support, I learnt from them a lot, and I wish them all the best in their life. I mention here Josef, Sen, Romina, Luise, Stefano, Housam, Jose, Aude, Vinnie, Lorenz, Yaman, Anna, Nikola, Nawar and Balint.

Finally, I dedicate this thesis to my awesome family. I thank them for believing in me, and for their unlimited source of love, inspiration, and motivation. I wish that Syria come to peace very soon, so we can retrieve our nice life together.

## Table of Contents

<b>List of Figures.....</b>	<b>IV</b>
<b>List of Tables.....</b>	<b>VII</b>
<b>List of Abbreviations.....</b>	<b>X</b>
<b>Chapter 1: Introduction and Motivation.....</b>	<b>1</b>
<b>Chapter 2: Current State of Research and Technology.....</b>	<b>4</b>
2.1 Localization and Navigation of Indoor Mobile Robots.....	4
2.2 Human-Robot Interaction.....	12
2.3 Collision Avoidance for Indoor Mobile Robots.....	20
<b>Chapter 3: Goals of the Dissertation and Realization Concepts.....</b>	<b>31</b>
3.1 Background and Work Description.....	31
3.2 Improving the localization of indoor mobile Robots.....	34
3.3 Human-Robot Interaction.....	35
3.4 Collision Avoidance for Indoor Mobile Robots.....	40
<b>Chapter 4: Localization of Indoor Mobile Robots.....</b>	<b>42</b>
4.1 Introduction .....	42
4.2 Improving the StarGazer Localization using Kalman Filter.....	43
4.3 The Improved Kalman Filter.....	46
4.4 Experimental Results.....	47
<b>Chapter 5: Human-Robot Interaction System for Indoor Mobile Robots.....</b>	<b>50</b>
5.1 Introduction.....	50
5.2 System Description.....	51
5.3 Support Vector Machine.....	53
5.3.1 Model Description.....	53
5.3.2 Model Training.....	56

5.4 Back Propagation Neural Network.....	57
5.4.1 Model Description.....	57
5.4.2 Model Training.....	60
5.5 System Implementation.....	60
5.5.1 Kinect Position.....	60
5.5.2 Human-Robot Interaction System Description.....	60
5.5.3 Sensor False Inferred Data.....	62
5.6 Experimental Results.....	63
5.6.1 Training the SVM.....	63
5.6.2 Training the BPNN Model.....	65
5.6.3 Comparison between SVM and BPNN.....	66
5.6.4 Human-Robot Interaction System Test.....	67
<b>Chapter 6: Collision Avoidance System for Indoor Mobile Robots Basing on Human-Robot Interaction.....</b>	<b>70</b>
6.1 Introduction.....	70
6.2 System Description.....	72
6.3 Collision Avoidance System.....	72
6.3.1. Cooperative Collision Avoidance based on Human-Robot Interaction.....	76
6.3.2. Autonomous Collision Avoidance.....	79
6.4 Collision Avoidance Path Calculation.....	80
6.5 Robot's Linear and Angular Velocities Calculation.....	82
6.6 Software Implementation.....	83
6.6.1 Development Tools.....	83
6.6.2 System Realization.....	83

6.6.2.1 Collision Avoidance Controller.....	86
6.6.2.2. Communication with the MFS.....	87
6.6.2.3 Collision Avoidance User Interface.....	88
6.7 Experimental Results.....	89
6.7.1 Tests over the Cooperative Collision Avoidance.....	89
6.7.1.1 Move Forward.....	89
6.7.1.2 Move Backward.....	90
6.7.1.3 Move Right.....	91
6.7.1.4 Move Left.....	93
6.7.2 Autonomous Collision Avoidance (ACA).....	94
6.7.3 Test of the Collision Avoidance for different Situations.....	95
6.8 Discussion.....	100
<b>Chapter 7: Conclusion and Outlook.....</b>	<b>101</b>
7.1 Conclusion.....	101
7.2 Outlook.....	103
<b>REFERENCES.....</b>	<b>104</b>
<b>Appendix 1: The StarGazer Sensor.....</b>	<b>113</b>
<b>Appendix 2: The Kinect 2.0.....</b>	<b>114</b>
<b>Appendix 3: The complete experiments for the SVM model.....</b>	<b>115</b>
<b>Appendix4: The experiments for the human-robot interaction system.....</b>	<b>117</b>
<b>Appendix5: The programming code for “Autonomous Collision Avoidance” function.....</b>	<b>120</b>
<b>Appendix6: The experiments for the collision avoidance system.....</b>	<b>125</b>

## List of Figures

Figure 2.1: Localization based on multiple ultrasonic sensors.....	5
Figure 2.2: The combined 2D maps from different reference nodes.....	7
Figure 2.3: The localization error in hybrid system.....	7
Figure 2.4: Comparison of the localization using odometer, and EKF.....	9
Figure 2.5: The filtration result of applying (a) extended Kalman filter, (b) the new filter.....	10
Figure 2.6: Assisting the robot to stand via interaction.....	13
Figure 2.7: The human-robot tracking and interaction system.....	14
Figure 2.8: The experiment environment for controlling the robot arm to reach the four locations.....	15
Figure 2.9: The integration between the Human and NAO robot based on gestures.....	17
Figure 2.10: Extracted velocity vectors of the action “sitting”.....	18
Figure 2.11: The motion imitation of the user’s arms by the robotics arms.....	18
Figure 2.12: The motion of the robot for the Bug1, Bug2, and Maze path.....	22
Figure 2.13: The representation of velocity space in DWA.....	22
Figure 2.14: The general concept of potential field method.....	24
Figure 2.15: The polar representation of the VFH and the direction selection.....	24
Figure 2.16: The representation of obstacles, gaps, regions, and valley in nearness diagram.....	25
Figure 2.17: The concept of implementing collision-free path using SND method.....	26
Figure 2.18: The vanishing point-based navigation.....	27
Figure 3.1: The integration of robotic tasks into the H20 robot.....	32
Figure 3.2: The structure of H20 Mobile Robot.....	33

---

Figure 3.3: False detection of landmark.....	35
Figure 3.4: The conflict in generating collision-free path between the robot and the human.....	37
Figure 4.1: The principle of StarGazer Localization sensor.....	43
Figure 4.2: StarGazer sensor and the passive Landmarks.....	43
Figure 4.3: The navigation under strong natural and fluorescent lights.....	44
Figure 4.4: The flow chart for the proposed filter.....	47
Figure 4.5: The StarGazer measurements before and after filtering.....	49
Figure 5.1: The gestures used for the interaction with the robot.....	53
Figure 5.2: The Principle of Support Vector Machines.....	54
Figure 5.3: The flow chart for building and training the L-SVM model.....	57
Figure 5.4: The k-fold sets for the L-SVM model.....	57
Figure 5.5: The hierarchy of the BPNN.....	58
Figure 5.6: Position of Kinect sensor and detection range.....	61
Figure 5.7: The control of the robot using Human-Robot Interaction.....	62
Figure 5.8: The false skeletal joints due to angle view limits of Kinect.....	63
Figure 5.9: Representation of the test environment (dimensions in meter).....	67
Figure 6.1: Robot motion between a group of humans.....	71
Figure 6.2: The flow chart of the collision avoidance system.....	73
Figure 6.3: Calculation of middle region.....	75
Figure 6.4: Calculation of the terminal region.....	75
Figure 6.5: The bottleneck problem when the robot and human are moving in narrow places.....	77
Figure 6.6: Cooperative Collision Avoidance via Interaction.....	78
Figure 6.7: The flow chart of Autonomous Collision Avoidance.....	79
Figure 6.8: Autonomous search for the collision-free path.....	80



Figure 6.9: Collision-free path calculation between two humans.....	80
Figure 6.10: The generation of CA path for the terminal person .....	81
Figure 6.11: The linear and angular velocities for different width of regions.....	83
Figure 6.12: The general architecture of the transportation system.....	84
Figure 6.13: The general architecture of the Cooperative Collision Avoidance System.....	85
Figure 6.14: The flow chart of collision avoidance controller.....	86
Figure 6.15: The XML messages between CA and MFS.....	87
Figure 6.16: The interface of the collision avoidance system.....	88
Figure 6.17: The experiment for “move forward” function.....	89
Figure 6.18: The experiment for “move backward” function.....	90
Figure 6.19: The experiments for the “Move Right” function.....	92
Figure 6.20: The experiments for the “Move left” function.....	93
Figure 6.21: The experiments for “autonomous collision avoidance”.....	94
Figure 6.22: The followed path in the first experiment.....	96
Figure 6.23: The motion of the robot for three locations.....	97
Figure 6.24: The followed path in the second experiment.....	98
Figure 6.25: The motion of the robot in the three locations for the second experiment.....	99
Figure A1.A: The IR sensor from Hagisonic- South Korea.....	113
Figure A1.B: HL2 Landmark with the hexadecimal values and the best distribution.....	113
Figure A2.A: The Kinect V2 sensor.....	114

## List of Tables

Table 2.1: Localization error based on Ultrasonic sensors.....	5
Table 2.2: Comparison between different localization sensors.....	12
Table 2.3: The experimental results for robot arm control based on brain signals...	15
Table 2.4: The experimental results of training and testing the face recognition model.....	16
Table 2.5: Summary of different systems for human-robot interaction.....	19
Table 2.6: The comparison of different HRI methods based on Kinect and gesture recognition.....	20
Table 2.7: Comparison between collision avoidance systems.....	28
Table 2.8: Comparison between different kinds of collision-avoidance sensors.....	29
Table 4.1: The experimental results of the Kalman filter over the StarGazer sensor.....	48
Table 5.1 The gestures used in the HRI system and their corresponding function...	52
Table 5.2: The training results for the L-SVM model.....	64
Table 5.3: The outputs of the BPNN for the given gesture.....	65
Table 5.4: The result of training and testing the BPNN for several hidden neurons.....	66
Table 5.5: The summary of experiments over the HRI system.....	68
Table 6.1: Summary of the experiments for “move forward” function.....	90
Table 6.2: Summary of the experiments for “move backward” function.....	91
Table 6.3: The experimental summary for the “Move Right” function.....	91
Table 6.4: The experimental summary for the “Move Left” function.....	94
Table 6.5: The experimental summary for the Autonomous Collision Avoidance...	95
Table 6.6: The experimental results for the first path.....	97
Table 6.7: The experimental results for the second path.....	98

Table A1.A: StarGazer Specifications.....	113
Table A2.A: Kinect 2.0 Specifications .....	114
Table A3.A: The complete experiments for the SVM model.....	115
Table A4.A: The experiments for the first person (height 165).....	117
Table A4.B: The experiments for the second person (height 170).....	118
Table A4.C: The experiments for the third person (height 177).....	118
Table A4.D: The experiments for the fourth person (height 179).....	119
Table A4.E: The experiments for the fifth person (height 188).....	119
Table A6.A: Experimental results for “move forward”.....	125
Table A6.B: Experimental results for “Move Backward”.....	126
Table A6.C: Experimental results for “move right” $v_{max} = 0.2 \text{ m/s}$ $\omega_{max} = 0.4 \text{ rad/s}$ .....	127
Table A6.D: Experimental results for “move right” $v_{max} = 0.25 \text{ m/s}$ $\omega_{max} = 0.3 \text{ rad/s}$ .....	127
Table A6.E: Experimental results for “move right” $v_{max} = 0.3 \text{ m/s}$ $\omega_{max} = 0.5 \text{ rad/s}$ .....	128
Table A6.F: Experimental results for “move left” $v_{max} = 0.15 \text{ m/s}$ $\omega_{max} = 0.3 \text{ rad/s}$ .....	128
Table A6.G: Experimental results for “move left” $v_{max} = 0.2 \text{ m/s}$ $\omega_{max} = 0.4 \text{ rad/s}$ .....	129
Table A6.H: Experimental results for “move left” $v_{max} = 0.3 \text{ m/s}$ $\omega_{max} = 0.25 \text{ rad/s}$ .....	129
Table A6.I: Experimental results for Autonomous CA $v_{max} = 0.15 \text{ m/s}$ $\omega_{max} = 0.2 \text{ rad/s}$ .....	130

Table A6.J: Experimental results for Autonomous CA

$v_{max} = 0.2 \text{ m/s}$     $\omega_{max} = 0.5 \text{ rad/s}$  .....130

Table A6.K: Experimental results for Autonomous

$v_{max} = 0.3 \text{ m/s}$     $\omega_{max} = 0.25 \text{ rad/s}$  .....131

## List of Abbreviations

ACA	Autonomous Collision Avoidance
API	Application Programming Interface
BPNN	Back Propagation Neural Network
CA	Collision Avoidance
CCA	Cooperative Collision Avoidance
CG	Closest Gap
DOF	Degree Of Freedom
DTW	Dynamic Time Warping
DWA	Dynamic Windows Approach
EKF	Extended Kalman Filter algorithm
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HOVV	Histogram of Oriented Velocity Vectors
HRI	Human-Robot Interaction
IFR	International Federation of Robotics
IMU	Inertial Measurement Unit
IR	Infrared
L_SVM	Linear Support Vector Machine
LM	Landmarks
LRF	Laser Range Finder
MAE	Mean Absolute Error
MbICP	Metric-based Iterative Closest Point
MFS	Multi Floor Navigation System
ND	Nearness Diagram

OAL	Obstacle Avoidance Layer
OL	Orientation Layer
PCA	Principal Component Analysis
PF	Particle Filter
PID	Proportional-Integral-Derivative controller
PIR	Passive Infrared Sensors
RANSAC	Random Sample Consensus
RRC	Robot Remote Centre
RSSI	Received Signal Strength Indicator
SLAM	Simultaneous Localization And Mapping
SND	Smooth Nearness Diagram
SUFR	Speed Up Robust Features
SVM	Support Vector Machine
UKF	Unscented Kalman Filter
VFH	Vector field histogram
VP	Vanishing Point
XML	Extensible Markup Language

## Chapter 1

### Introduction and Motivation

---

The motivation of building machines which are able to understand the human, help them in their daily life and work in a same way as humans do is dated back to hundreds of years [1],[2]. This motivation is kept unachievable due to the lack of the advanced technology, which is needed to create such machines, till the middle of the 20<sup>th</sup> century. **George Devol** designed the first programmable robotic arm and sold it to General Motors in 1960; while Elmer robot (1949) could be considered as the first mobile robot, which is able to avoid obstacles and move to charging station. Since that time, the robotics made great strides; thanks to the powerful processors, artificial intelligence, sensory systems and many other fields which offered promising applications for the mobile robotics in daily life. Nowadays, the world is on the door of the fourth industrial revolution which robotics is one of the fundamental pillars in it. That's why, the major companies and research institutes are investing intensively in this field. According to the International Federation of Robotics (IFR), the total number of sold service robots in 2014 is 24,207 [3], and this proves that the robotics technology brings promising solution in the close future.

Many robotic systems are put into investment, as robotic arms, quadrotors, drones, indoor mobile robots, surgery, and space robots. The fast progress in mechanical engineering, electric engineering, computer systems, and artificial intelligence will lead to have robots in many other fields such as self-driving cars, military, industry and life science laboratories.

In life science laboratories, preparing and handling of chemical and biological samples need long times, and it requires 24 hours monitoring and processing for these samples. In recent years, progressively automation systems have been applied in chemical & biological laboratories [4-7]. These advances brought many facilities, and decreased the processing time and number of employees.

Current research is ongoing on the development of fully automated chemical, biological or analytical laboratories with minor human supervision [8-11]. To ensure a complete automation, mobile robots are used as transport systems for the transfer of samples and lab ware between different automated and manual islands. Recent results include several steps toward having a complete autonomous transportation

system based on mobile robotics, such as multi floor navigation [12], arm manipulation [13], charging [14], and path planning [15].

Mobile Robots are advanced engineering systems, which require a high integration of electronic engineering, mechanical engineering, computer systems, sensory systems and artificial intelligence. Robots must be equipped with several sub-systems to work securely in the real applications, such as path planning, localization and mapping, human-robot interaction, manipulation, collision avoidance, and charging. The good implementation of these sub-systems, and the integration between each other is the key-success for any mobile robotic system.

Unlike traditional automation and control systems, mobile robots will work in unpredicted environments; i.e. the work conditions are changing so there are no certain situations that the robot could be pre-programmed to adopt its control to meet these situations. Taking into consideration that the robots will navigate nearby human, this raises further challenges regarding the ability of the robots from detecting the human and avoiding physical injury to him.

The main requirement for any engineering system is safety. In mobile robotics, safety means the ability of the robots from avoiding accidents which could occur when the robot loses its location within the map or when it collides with the obstacles which might appear in its path.

The ability of the robots for a secure navigation in their work area is highly required taking into consideration that in future life science laboratories, robots must navigate in narrow paths, inside rooms with small free spaces. Thus, a robust localization system is necessary to ensure that these robots are able to identify their positions despite of the external noise over the sensors. If a robot fails to accurately identify its position within the map, it is too possible that it collides with walls or doors, or loses the positions of grabbing, charging, and elevators. Furthermore, the narrow corridors and the small free spaces increase the strains of having a robust localization system.

The robots and the human will work in the same area. Thus, the robots must detect the human, interact with them and avoid the collision when both are located in the same place. Thus, a robust human-robot interaction is necessary to enable the robots from detecting the human and communicate with them. Furthermore, the robots must be able to generate local paths to avoid the collision with those human and keep moving to their destinations.

The aim of this dissertation is the development of suitable concepts for indoor mobile robots to eliminate failures due to false localization in the work area, or due



to the collision with humans, which are located near to the robots. This work can be divided into three parts basing on the processed problem in mobile robots:

- Implementing a robust localization system for the indoor mobile robots, by surveying the relevant sensory systems which fit the requirements of laboratory environments, evaluate the performance of the selected sensors and find the suitable methods to improve the performance of the localization system. The StarGazer sensor is utilized for mapping and localization. The sensor is tested under different conditions, and a modified Kalman filter algorithm is used to eliminate the noisy measurements resulting from strong lights. The localization system is discussed in chapter 4.
- Implementing a human-robot interaction system based on gesture recognition to enable the robot to recognize the humans and interact with the orders given by them via certain movements or the arms. The robot uses 3D vision to recognize the human, and Back Propagation Neural Network model for processing the data of the sensor. The interaction is used mainly to coordinate the generation of the collision-free paths. Further details for the human-robot interaction system are available in chapter 5.
- Designing a robust collision avoidance system for avoiding the humans existing in the path of the robot. The system has to detect the human, calculate the location of each person from the robot, and generate a new local collision-free path to avoid accidents as it will be described in chapter 6.

The following thesis is organized as follows: *Chapter 2* presents the current state of art in the field of secure navigation and operations of indoor mobile robots. This includes a comprehensive survey over current applications of the indoor mobile robots, a review for the recent work of the localization and navigation systems, human-robot interaction in mobile robotics, and a review for the collision avoidance systems for indoor mobile robots. *Chapter 3* discusses the goals of the dissertation and realization concepts. *Chapter 4* shows a framework for improving the localization of indoor mobile robots. *Chapter 5* details the implementation of the human-robot interaction system. In *chapter 6*, the new method of collision avoidance based on the interaction between the human and the robot is proposed. Finally, *chapter 7* shows the conclusion of the implemented work.

## Chapter 2

### Current State of Research and Technology

---

Mobile robots are machines, which are able to move in the navigation area and modify their behaviour basing on the changes in the work conditions without the need of human interventions. These robots could implement their tasks either in small areas as homes and factories, or in vast areas as the space or deep sea robotics. Several research groups are working on inventing mobile robots which are appropriate for certain tasks, as hospitals [16], cleaning [17], material handling [18], agriculture [19], and outdoor autonomous robots [20]. Each of these robots needs sensory systems for localization and identifying the working environment.

#### 2.1 Localization and Navigation of Indoor Mobile Robots

Different localization systems can be used in mobile robotics to serve the task of indoor localization and mapping. These systems use different kinds of sensors which work on providing the localization information to the robot, such as ultrasonic sensors, wireless sensors, vision sensors, and IR sensors. This section shows the common sensors and localization systems which are used for indoor mobile robots. Each of these methods have its advantages and limitations.

Ultrasonic sensors have the advantages of linear performance, cheap cost, and it is not effected by light noise. Still, these sensors suffer from the angle limitation problem [21] since the transmitter issues ultrasonic waves with a limited angel, and this limits the width of the sensed area, besides to noise effects caused by sound waves. To compensate the limitations of this technology and overcome the noise effects, a sensory fusion is used with the other kind of sensors.

**Kim et al.** [22] used a group of ultrasonic transmitters/receivers for robot localization. A group of ultrasonic transmitters (beacons) were fixed in known positions, and three ultrasonic receivers were fixed on the top of the robot. The receivers measure the floating time of the received signal from transmitters to estimate the robot's location. Extended Kalman Filter algorithm (EKF) fuses the measurements obtained from the encoder (in prediction phase), and ultrasonic sensors (correction phase) to obtain optimal estimation of the robot position. Fig 2.1

shows the robot localization basing on distributed ultrasonic sensors. The performance of this system is merely based on the efficient distributions of the beacons in the navigation area. The localization system fails when the robot is located in not covered areas. Furthermore, the experiments show that the localization error could reach to 25.7 cm and  $6^\circ$ .

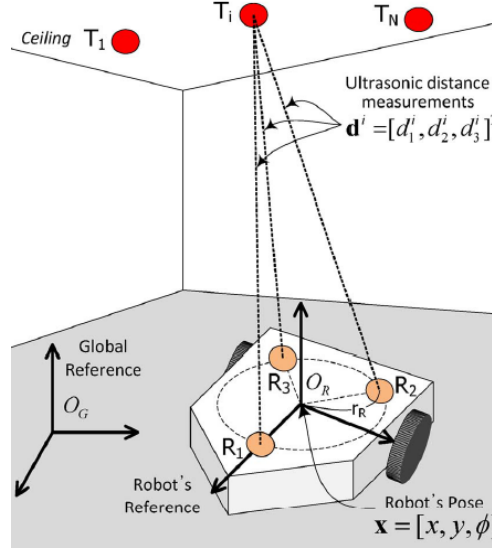


Figure 2.1: Localization based on multiple ultrasonic sensors [22]

**Alfonso et al.** presented a comparison of EKF and unscented Kalman filter (UKF) in estimating the localization of a mobile robot using a group of five ultrasonic sensors [23]. The prediction phase of the filter is done basing on the readings of encoders. The update step is done basing on the measurements of ultrasonic sensors. The study focused on improving the localization basing on switching the ultrasonic sensors for saving the battery energy. Table 2.1 shows the localization error for each sensor (S1 to S5), the localization error of switching these sensors, and the localization error of operating the whole sensors together (all).

Table 2.1: Localization error based on Ultrasonic sensors [23]

Filter	S1	S2	S3	S4	S5	Switching	all
EKF							
$\eta_t=0$	4.354	6.698	5.056	4.964	4.784	6.007	3.358
$\eta_t=0.3$	4.177	6.472	4.948	4.888	4.21	6.041	3.215
$\eta_t=0.7$	4.42	6.1	4.734	4.628	4.037	6.088	3.298
UKF							
$\eta_t=0$	4.301	5.849	5.272	4.892	4.389	5.098	3.432
$\eta_t=0.3$	4.129	5.544	4.838	4.647	3.830	5.446	3.358
$\eta_t=0.7$	4.244	5.585	4.546	4.517	3.669	5.184	3.179

$\eta$  represents the ratio of lost measurement ( $\eta=0.3$  means 30% of measurements were lost). Experiments show that both UKF and EKF have close performance, and the proposed method provides a localization error around 3.2 cm, when the whole sensors are operated together [24 - 26].

**Dobrev et al.** built a localization system for indoor mobile robots by using a combination of radar, ultrasonic, and encoder sensors [27]. The radar localization system is composed of a reference node, and a mobile node which is located on the robot. This sensor works well in detecting the location of the robot in halls and foyers, but the performance is degraded when the robot moves in corridors because the radar waves will reflect from the walls and this cause to lose the robot's location. Thus, the ultrasonic sensors work on improving the performance of radar sensor by measuring the distance of the robot from the side walls of the corridor and corrects the localization latency of the radar sensor. The measurements of the radar and ultrasonic sensors are then fused with the encoder by using extended Kalman filter. In the first experiment, the robot is navigated in wide hall, and the experiment shows that the measurements of radar sensor is unreliable when the distance between the reference and mobile nodes is more than 15m. Furthermore, the localization accuracy is degraded when the robot moves in corridors by only using the radar sensor. On the other hand, when using the integrated system, the experiments show that 68.3% of the measurements have a localization error less than 5.1cm when the robot moves into corridors.

Wireless sensor networks can be seen as an infrastructure, which is composed of distributed sensing elements, and communication elements, which collect data and measurements in a real environment [28]. Zigbee is a wireless technology which uses the standard IEEE 802.15.4. This technology allows building wireless networks by using a set of Zigbee nodes over the work area. This technology could be used in the indoor localization of mobile robots.

**Lin et al.** developed a localization system based on a group of ZigBee nodes which compose a wireless network distributed in the navigation area of the robot, and a mobile node which is located over the robot [29]. The positions of the reference nodes are stored on the onboard PC. The mobile node broadcasts measurement-demand messages to all reference nodes; the reference nodes will then measure the received signal strength (RSSI) and send it back to the mobile node. The localization system is divided into two phases: calibration phase and localization phase. The calibration is implemented by finding the relationship between the received signal strength indicator RSSI and the distance of the mobile node from each reference node. A 2D probability map is realized which plots the RSSI readings for different distances for each reference point. In the localization phase, the location of the

mobile node is estimated by combining the 2D maps of different reference nodes. Fig 2.2 shows the resulting augmented map with the estimation of the robot's location. The experimental results show that the localization average error is around 1.7m [30].

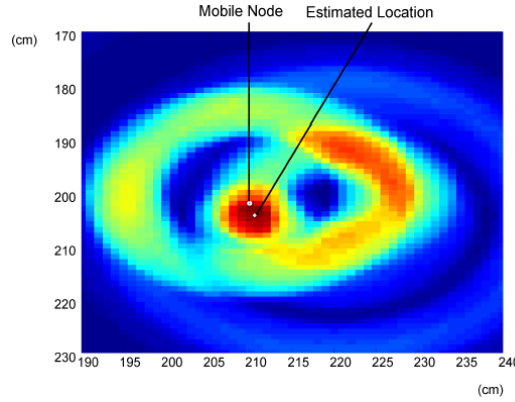


Figure 2.2: The combined 2D maps from different reference nodes [29]

**Alhmiedat et al.** presented a hybrid localization system basing on the measurements of received signal strength indicator (RSSI) and Inertia system [31]. A wireless sensor network based on the ZigBee standard is used to obtain the measurements of distributed RSSI. An onboard inertia system is composed of an acceleration sensor and compass, which specify the direction and the speed of the robot. A hybrid system is designed to integrate the RSSI and inertia systems. The results show that using a hybrid localization system basing on data from wireless sensors and inertia system is higher than using RSSI alone. Fig 2.3 shows the localization error using the hybrid system. It can be seen that the localization error is around 0.6 meters using the hybrid system [32] - [34].

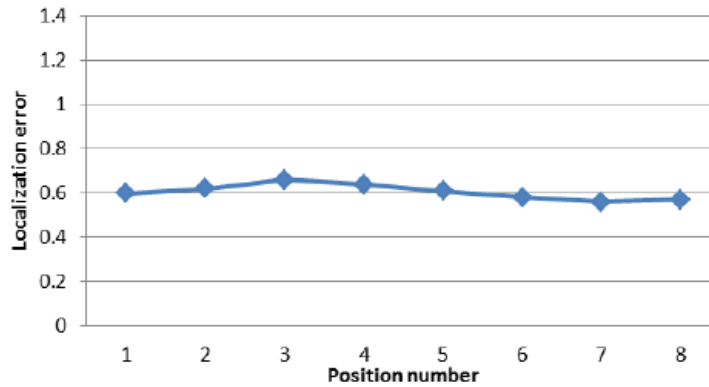


Figure 2.3: The localization error in hybrid system [31]

Vision sensors are popular in mobile robotics. Common vision sensors in mobile robotics are Colour – depth cameras (RGB-D) [35], [36] and omnidirectional cameras [37].

**Fernandez et al.** implemented a simultaneous localization and mapping system (SLAM) for indoor mobile robot by processing the images provided by a visual odometry [38]. In this system, the robot moves in unknown environment, and it has to build a topological map for the area by capturing certain images which will represent the reference nodes in the map. To describe each image, a Fourier signature descriptor is employed. Each node represents a certain area, and contains certain images describing it. After building the map, the robot can find its position by comparing the descriptors of the current image with the descriptors of the images stored in the database. Monte-Carlo algorithm is used to combine the visual localisation based on the topological map with the robot's odometry to create a visual odometry. The experimental results show that the robot requires 200 to 2000 particles at each step of the robot to guarantee that it can always detect its position within the map, and the computational cost is equal to 0.223s and 0.243s respectively. [39], [40].

In localization based on laser range finder (LRF), the robot uses a laser sensor to draw a map for the environment and specify its position within the map. Moreover, the LRF could be used in human detection and obstacle avoidance [41].

**Liu et al.** presented a new method for indoor map building and localization using 2D laser scanner [42]. To build the map of the working area, the robot implements a primary scan for the room to find the center of the location; and it will then rotate 360° in the center to implement a complete 2D scan for the room and build the map. To reduce the processing time, the robot uses grid-based algorithm to remove the redundant points, and then it will use least-square algorithm to extract the lines and the boundaries in the room. After building the map, the robot uses metric-based Iterative closest point (MbICP) algorithm to compare its current scan with the built map and derive its translational and rotational displacement. Furthermore, EKF is used to estimate the location of the robot. The EKF uses the measurements from the odometer as a control input, and the observation from the LRF to estimate the location. The experimental results for mapping and localizing using LRF and EKF show that localization based on EKF is better than using dead-reckoning alone. Fig 2.4 shows the improved navigation results with the proposed system, compared to the navigation with odometer sensor; it can be seen that the accumulative error in the odometer will cause the robot to deviate from the path for around 30cm on Y axis, while the navigation with the implemented system doesn't suffer from the accumulated error and it still has accurate localization [43], [44].

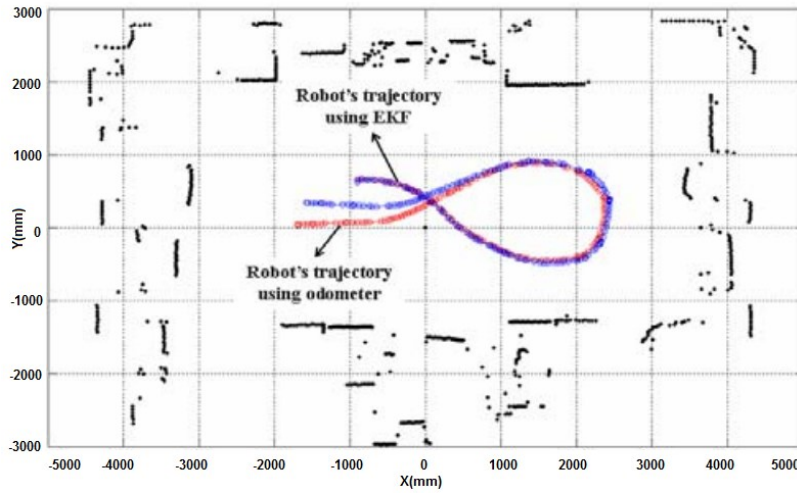


Figure 2.4: Comparison of the localization using odometer (red), and EKF (blue) [42]

In localization based on Infrared (IR) landmarks, the robot identifies its location based on a group of passive Landmarks (LM). These landmarks are composed of IR reflecting materials. An IR camera analyses the reflected IR light and specifies the location based on this analysis[45], [46].

**Zhiwei et al.** presented the use of the StarGazer sensor for the localization and mapping of indoor mobile robots [47]. This sensor is composed of IR emitters and an IR camera installed on a single board. The emitters release the IR beam to the passive landmarks, and the reflected beam is captured by the IR camera. The study shows that the sensor provides some false detection due to the noise which results from strong light, and this causes the robot to lose its location. To overcome this limitation, three filters are implemented and the performance is compared: extended Kalman filter (EKF), particle filter (PF), and a new method which is based on defining an error range for the sensor. The proposed method considers that the false detection of the landmarks couldn't be represented as a white Gaussian noise. Thus, the PF and EKF won't show satisfactory results. In the proposed filter, the measurement error is defined within a certain range rather than considering it as Gaussian; thus, if a new measurement is within the estimated measurement  $\pm$  the error, the measurement is considered true, else it will be considered false. Moreover, the filter will implement a sensor fusion for the local data of the relative position of the robot (encoder), and the global data obtained from the StarGazer. Fig 2.5 shows the results of applying extended Kalman filter (EKF), and the proposed filter over the measurements. It could be seen that the EKF failed to detect the false measurements with error  $> 3$  m. On the other hand, the new filter was able to detect

the false measurements with error  $>3\text{m}$ , and provide an accurate estimation for the location of the robot. [48 - 54].

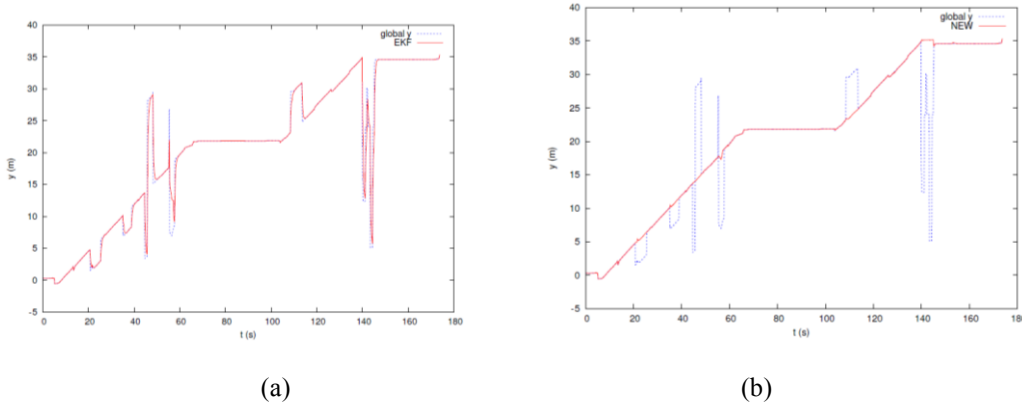


Figure 2.5: The filtration result of applying (a) extended Kalman filter, (b) the new filter. Blue dots represent the raw measurements of StarGazer sensor, and the red dots represents the filter output [47]

In summary, many localization and mapping methods and sensors are proposed in the field of indoor mobile robots; each has its advantages and limitations. In localization and mapping using ultrasonic technology, it shows fast processing time, besides to its cheap cost. On the other hand, this technology requires a high number of distributed transmitters due to its limited rang, and the landmarks are active, which means there is a need to power each transmitter/receiver. Furthermore, modifying the map will be more difficult comparing to the maps which use natural landmarks [22].

Localization based on wireless sensor network and ZigBee still needs further developments to cope the problems of fast fading in which the signal strength is decreased significantly due to the existence of walls, and multi paths, especially in indoor environments which include walls [55]. Furthermore, this kind of localization has a low localization accuracy [29],[31], and this is not suitable for the tasks which require accurate localization as in locations of grasping elements, and entering narrow location as elevators.

Vision sensors provide rich information to the robot which could be used in localization and mapping, obstacle avoidance, and human detection. Alternatively, these sensors have high computational cost, besides to its affect to the changing in the light conditions in the navigation area.

Localization based on LRF is an accurate technology; it could be used for mapping, localization and obstacle avoidance. Still, this technology requires high



computational costs since it requires processing a high amount of measured points over the work area [42]. Furthermore, the laser sensor is not suitable to work in environments, which include transparent surfaces like glass and plastic since these materials could diffuse the beam rather than reflecting it [44], [56].

Localization based on IR sensors has the advantages of covering a wide area, uses passive landmarks without the need to power them and it is not affected by radio transmission. On the other hand, high fluorescent light or sunshine could cause false detection of the landmark, and this will lead the robot to lose its location.

Table 2.2 summarizes the advantages and disadvantages of each method, and the applicability of using this sensor for mobile robotics in life science applications.

Table 2.2: Comparison between different localization sensors

Sensor Type	Advantages	Disadvantages
Encoder	Easy to use Linearity	The accumulated error limits the efficiency of this sensor for long distances
Ultrasonic	Easy to use, Linear, Low cost	Need to power the transmitter and receiver, difficult to modify the map Effected by changes in humidity and temperature
Wireless and RSSI	Data transmission could be implemented together with sensing, no need for change the direction of sensor to measure the distance	Not constant, the existence of walls effects the accuracy of this system
Vision	Low cost, Implement localization and obstacle detection in one sensor	High computation, Affected by light conditions
Laser	High Accuracy Used for several tasks in one sensor	Lower performance in detecting transparent materials such as glass walls.
Stargazer	Easy to use Map could be extended easily No need for powering the landmarks Automatic height calibration	Affected by ambient light such as strong sunlight and fluorescent light.

## 2.2 Human-Robot Interaction

Human-Robot Interaction (HRI) is defined as the ability of a robot for recognizing the human and interacting with them by implementing suitable responses to this interaction. Several sensory systems and interaction methods are implemented for

the interaction; interaction systems are based on the type of the robot, and the interaction method.

**Ikemoto et al.** presented a physical human-robot interaction system for teaching the robot to stand up [57]. The human will help the robot in standing-up, and he will inform it whether the attempt was correct or wrong. The robot consequently will save the correct attempts in its database to train itself in a later step. Each training vector is 52-dimensions which represent the angular values of robot's joints. To accelerate the training process, principal component analysis is used for dimensions' reduction of the training set. The reduced data is passed to a Gaussian mixture model (GMM) for adapting the performance of the robot in the standing-up task. The experimental results show that reducing the training data allows the GMM model to be trained online with few training data, while the robot develops its behaviour after each training step. Fig 2.6 shows an experiment for the interaction system.



Figure 2.6: Assisting the robot to stand via interaction [57]

Wearable sensors are another common method for implementing the interaction between the human and the robot. **Cifuentes et al.** developed a human-following system to track the human using LRF and a wearable inertial measurement unit (IMU) [58]. Instead of following the human, the robot will move in front of the human, enabling the human to monitor the motion of the robot without the need to look behind. LRF is located over the robot, and it is used to track the legs position and orientation of the human, while the IMU unit is fixed on the trunk of the person to provide the trunk motion and rotation to the robot. The robot will adjust its motion direction and velocity based on the obtained measurements to keep itself moving in front of the human. The experimental results show that the parameters estimation was accurate with estimation error less than 10% under curve-shaped path.

**Tseng et al.** presented a robotic system for tracking one person or a group of human and interacting with him/them [59]. In this system, the robot will differentiate between seven situations; basing on whether there is one person, or several humans in front of it. Furthermore, the robot will analyze the distances between the human, and their orientations toward each other. For example, when the robot detects human

facing each other and having a discussion, the robot will come close to them, and interact with them using voice commands to communicate available services as it could be seen in Fig 2.7. In contrary, when two humans are close to each other, and located in certain angles, the robot will know that these two human have private discussion, so it won't interrupt them. The system uses LRF for detecting and tracking legs, high resolution color camera for face detection from far distances, and depth camera for torso tracking. To decrease the processing time of face detection, Haar feature based cascade classifier is used to detect the upper body, and then a particle filter is used for face tracking. Furthermore, Kalman filter is used to track the position of each person. The positions of humans, and their orientations are analyzed using F-formation to receive the social cues. A decision tree will be used by the robot to select the correct response. The experimental results show that the average time to identify the social situation for a groups is 4.78 seconds, and the accuracy ranges between 80-90%.



Figure 2.7: The human-robot tracking and interaction system. The robot-to-group situation, the robot specifies that the social situation is normal discussion, the robot will come closer and interrupt the discussion asking for a service to do by itself [59]

**Hortal et al.** implemented a brain-machine interface for enabling a user to control a robotic arm using brain signals without the need to implement any physical activity by the user [60]. The task is to control the robot arm to move forward, backward, right, and left. The robot has to move to the right/left when the user thinks in moving his right/left arm respectively; it will move forward when the user counts down, and backward when the user thinks in alphabet in backward manner. The brain signals are obtained using a wearable cap which has 16 channels to harvest the signals from the scalp of the user. After amplifying and filtering the signals, it is forwarded to a support vector machine model to classify the activities. The SVM is trained offline first, and a k-fold algorithm is used to train the SVM and evaluate the model before employing it online. To test the system, two users were asked to move the robot between four points sequentially as it could be seen in Fig 2.8 and the success rate is then averaged as it could be seen from table 2.3 which shows the mean success rate for the two users are 74%, 72.78%.

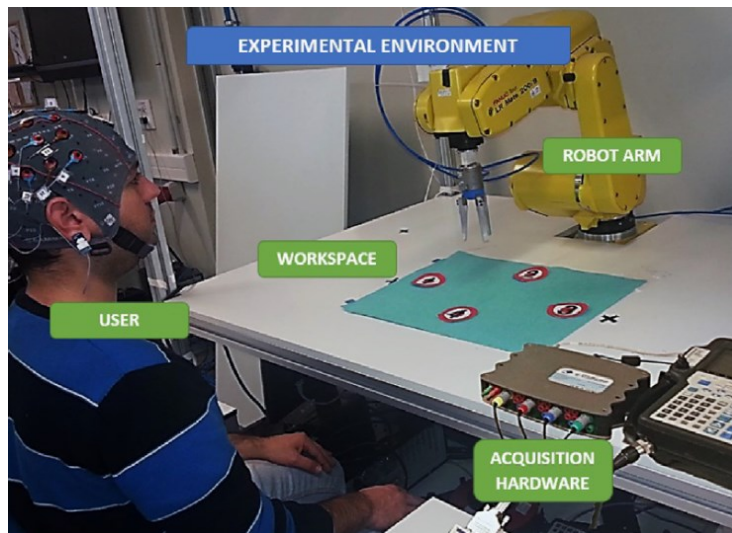


Figure 2.8: The experiment environment for controlling the robot arm to reach the four locations [60]

Table 2.3: The experimental results for robot arm control based on brain signals [60]

User	RH	LH	CD	AB	Mean
A	85.00	91.10	51.27	68.64	74.00
B	78.39	89.41	61.86	61.44	72.78

**Tsai et al.** presented a human-robot interaction system for face identification, and emotion recognition for four situations: sad, happy, anger, and smile [61]. The training samples, which represent the face images for different human are reduced using the Harr wavelet transform. The principal component analysis (PCA) is used to extract the features of faces from the images. While the Euclidean distance method is applied to measure the shortest distance between the training and testing samples so the processing time is reduced. A support vector machine mode (SVM) is then trained to classify the facial expressions of the user. The experimental results show that when the model is trained with a higher number of training samples, the training will take more time, but the model will be able after training to define the face and classify the emotions with a shorter time and higher accuracy. Table 2.4 shows two experiments for two SVM models, the first model is trained with 120 samples, while the second model is trained using 240 sample. The first experiment shows the face identification and emotion recognition success rates for the trained SVM with the processing time when the model is trained with 120 samples, and the second experiment shows the success rate and processing time for the SVM model when it is trained using 240 samples. It could be seen that the SVM model which is

trained with a higher number of training samples (240) has a higher accuracy and less processing time than the SVM model which is trained only with (120) samples.

Table 2.4: The experimental results of training and testing the face recognition model [61]

Number of training samples	Face identification	Processing time	expression recognition	Processing time
120	90%	5.73 p/sec	84%	6.9 p/sec
240	92%	5.5 p/sec	89%	6.5 p/sec

Voice recognition is another common method for the interaction between the human and the robot. **Ding et al.** developed a voice and user recognition system to interact with humanoid robots using 10 commands [62]. A support vector machine model is trained for speaker verification, a Gaussian mixture model is used for speaker identification, and a dynamic time warping algorithm (DTW) is used for speech recognition. When a user speaks, the SVM will decide whether the voice is from an invalid user, if the voice is from a valid one, the GMM in the next step will decide whether the user is authorized to control the robot or not. If the GMM shows that the user is authorized, the DTW will be used for the speech recognition. The system uses Kinect microphone array to obtain the voices. The experimental results show that the proposed system is able to recognize 85% of the voices successfully.

3D vision sensors are widely used nowadays in human-system interaction for gaming and robotics applications. **Canal et al.** used the Kinect 2.0 sensor in the interaction task between the human and humanoid robots via recognizing the gestures provided by the arms and the face of the human [63]. The system identifies four gestures: pointing at, waving, nodding and negation. The pointing is used to guide the robot to detect certain objects, which are identified with waving gestures by analyzing the angular displacements of the arm for certain frames, while the negation and nodding gestures are recognized by using the dynamic time wrapping approach for analyzing the face movements. The experiments were implemented over the NAO robot, with a wheeled platform which is used to carry the robot and the sensor to the goal destination as it could be seen in Fig 2.9. The experimental results show that the facial gestures have low recognition rates (33.33% for negate, 73.33% for nod) due to the misalignment of the face and the sensor plane, while the recognition rate for wave and point gestures are 83.33% and 96.67% respectively. The recognition requires processing times between 1.47 to 1.91 seconds.



Figure 2.9: The integration between the Human and NAO robot based on gestures [63]

**Yang et al.** presented the use of Kinect sensor in controlling NAO robot via gestures [64]. The first step is tracking the skeleton, and filtering the joint angle of the human controller. Kinematic analysis is used to transfer the human arm joints angles to control commands to the robot. Finally, a limit breadth method was applied to the system to protect the robot when the rotating angle of the controller exceeds the permissible rotating angle of the robot. The experimental results show that the robot is able to pass 92% of the experiments successfully.

**Wang et al.** presented a human-robot interaction system for controlling the Khebra III robot by identifying the gestures of human arms which are provided by a Kinect sensor [65]. The system defines 11 gestures from the human. The gesture recognition algorithm uses 6 of 20 joints and monitors the angles between joints for defining the gesture. The law of cosine is used for identifying the locations of joints and identifies the required response. The experimental results show that the system is able to successfully identify 96% of the experimental tests.

**BouBou et al.** implemented a new system for defining nine actions (sit, stand, wave, walk, pick up, stretch, use hammer, draw a circle, and forward punishing) [66]. The skeletal data of the tracked person is obtained from Kinect sensor, while a new method called Histogram of Oriented Velocity Vectors (HOVV) is implemented to describe the activities. Fig 2.10 shows the concept of the HOVV method. In HOVV, the velocity vector and orientation of each joint is extracted from tracking the 3D positions of the skeleton's joints obtained from Kinect. The obtained vectors are grouped into a spatial histogram. The histogram will be used to describe the actions. The robot uses the built method to analyze the human activities. The experimental results show that the system is able to classify the gestures with an average accuracy of 88.75% and with computational latency equal to 0.055 seconds for one sample. [67 - 73].



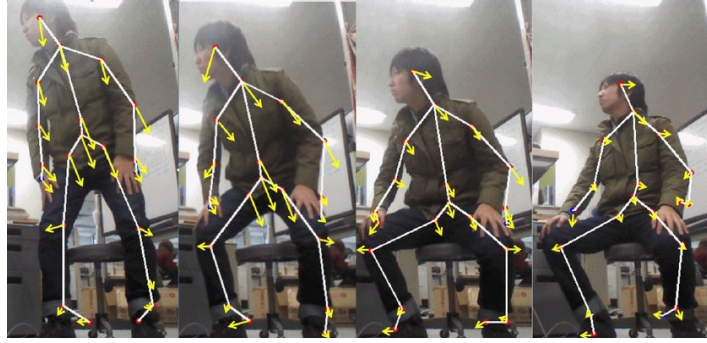


Figure 2.10: Extracted velocity vectors of the action “sitting” [66]

**Du et al.** presented a human-robot interaction system for controlling two robotic arms [74]. The robotic arms imitate the movements of the right and left arms of the human operator remotely. The system uses Kinect sensor to get the movements of the operator’s arms, specifically the thumb finger, index finger, wrist, elbow, and upper arm. The reference point of the user arms’ coordinates is the shoulder joint, so even if the user body moved, the robotic arms won’t be effected by this movement, and will consider only the movements of the arms related to the shoulder. Furthermore, the system took into consideration the dithering of the user hands’ by adding a damping model which ignores the minor movements of the user arms. Fig 2.11 shows the control of the robotic arms to catch an element without contact. The experimental results show that the mean absolute error (MAE) is (3.65, 2.67, 3.83 mm) and (1.01°, 1.17°, 1.5°) for (X, Y, Z) respectively.

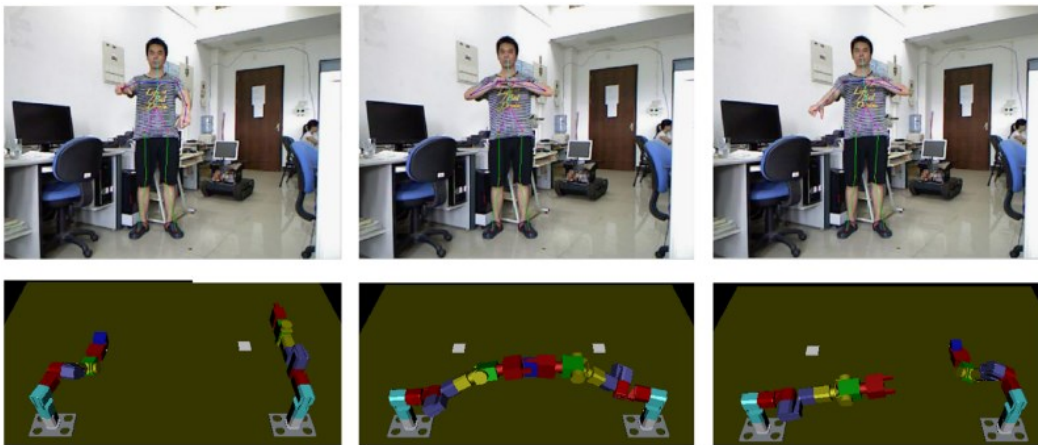


Figure 2.11: The motion imitation of the user’s arms by the robotics arms [74]

In conclusion, many systems are proposed for the interaction between the human and the robots existing in the same work area. Designing these systems is mainly



based on the application domain, and the goal of using the robot. Table 2.5 summarizes the common methods which are used in human-robot interaction.

Table 2.5: Summary of different systems for human-robot interaction

	Advantages	Disadvantages
Wearable Sensor	Obtain the measurements directly from the user. Not effected by noise resulted from light and sound	Not applicable when the robot works in social area since it needed to let each person to wear the device
Voice recognition	Easy and natural interaction method. The robot could be trained for a wide range of commands	Not suitable to detect humans on large distances. Can't provide more information as the distance of human(s). Not suitable in social environment in which include voice noises Can't distinguish the voice of different humans.
RGB	Provides rich information which could be used in other tasks as obstacle detection Cheap sensors	Requires high processing Additional landmarks could be required to distinguish the human
Kinect	Provides different HRI as voice, gesture, and face. Low cost, with low computational cost	Affected by the vibration which lead the sensor to lose the skeleton frames

In conclusion, the interaction using the wearable sensors isn't applicable in work environments which many human move in it as restaurants, and laboratories, since it requires a wearable device for each new person enters the work area. Moreover, the interaction with voice orders is easy and natural method which is used between human themselves. Still, this kind of interaction won't be suitable in work environments which include noise of machines and human, since it will limit the ability of the robot from recognizing the voice, and this will force the human to interact with the robot with short distances, and that won't serve the goal of this thesis to have a secure interaction. The interaction using color cameras won't be suitable when it is required to follow the gestures of several human, besides to the high processing time to analyze the images and define the human. On the other hand, the Kinect sensor has several advantages which serve the interaction between the human and the robot; it offers several possibilities for interaction, as voice, gesture and face interaction, besides to provide the 3d dimensions of the body.

Moreover, this sensor could be used for collision avoidance for mobile robotics. Thus, in this research, the Kinect sensor is adopted to serve the task of human-robot interaction.

Table 2.6 summarizes the common methods for the gesture recognition using the Kinect sensor. From the literature survey, it is noticed that the most methods which are followed in gesture analysis are related to monitoring the angles between the body's joints. These methods are suitable when the Kinect is fixed in static position, and facing the human face to face. In the real work environment, the robot will meet the human with different angle of views; the human could meet the robot face to face, or deviated with certain angles from the front plane of the Kinect sensor. Thus, the previous methods won't help, because the angles will be different. Thus, it is expected that using machine learning and artificial intelligence will improve the gesture detection if the models are trained with suitable training samples. Thus, two models are selected in this research (Support Vector Machine, and Back Propagation Neural Networks), and the performance of each model is analysed and compared. Moreover, the models were trained using samples with different angle of views, to enable the HRI system from defining the gesture even when the human and the robot are not located face to face.

Table 2.6: The comparison of different HRI methods based on Kinect and gesture recognition

Method	Success Rate
Analysis the angular displacement of the arms [63]	83.33% (waving) 96.67% (pointing at)
Extracting the joints' angles of the skeleton [64]	92%
Law of cosine [65]	96%
Histogram of Oriented Velocity Vectors [66]	88.75%
Hidden Markov Model [68]	98.4%

### 2.3 Collision Avoidance for Indoor Mobile Robots

Collision avoidance is a primary requirement for any robotic system. Many methods are used in mobile robotics for avoiding dynamic and static obstacles. Some methods generate obstacle-free path using control theory algorithms such as PID and Fuzzy Logic controllers. Others take into consideration the robot's dynamics. In contrary, reactive collision avoidance methods avoid the obstacles using the information provided by the robot's sensor only, without care about the robot's

dynamic. Here is a review for the common methods of collision avoidance, which could be found in mobile robotics:

Fuzzy logic is a control algorithm which is developed by Lotfi Zadeh (1965) to control the systems based on the “degree of truth” rather than the “crisp values” in which the variable values are either “0” or “1”[75]. This controller has been used in many obstacle avoidance systems.

**Lee et al.** used a fuzzy controller for navigation and obstacle avoidance in unknown environments using ultrasonic and compass sensors [76]. The system is divided into three layers: The orientation layer (OL) which controls the robot to reach the goal destination when no obstacles are existed in its path, the obstacle avoidance layer (OAL) which modifies the robot’s movement in the case of obstacles, and the human control layer which has the highest priority, and enables the human to control the robot. Each of OL and OAL represent a fuzzy controller, and the control output of the robot is the fusion of the outputs of the both layers. The human control layer represents a remote control panel which could be used by the human to control the robot, and it has the highest priority over the other two layers. The experiments show that when the path has no obstacles, the robot will be dominated by OL to reach the goal, when the robot faces an obstacle, the robot will start decreasing its speed and modifying its path due to the effect of the OAL layer, with the ability of the human to interrupt the movement and control the robot at any time [77], [78] [79].

Bug algorithm is one of the early work in the field of obstacle avoidance for mobile robotics [80]. Two algorithms are proposed: Bug1, and Bug2. In Bug1, the robot moves toward its goal location till it hits an obstacle. The robot will then implement a complete travel around the obstacle, parallel to its boundaries till it reaches again to the point that it hit the obstacle. If the goal is reached, the algorithm will stop, else the robot will select the leaving point which has the shortest distance to the goal. In Bug2 algorithm; instead of implementing a complete navigation around the obstacle, the robot will move in parallel to the obstacle till it meets again the straight line linking the starting point and goal point. It will then stop moving around the obstacle, and direct itself to follow the starting-goal line. The experiments show that each algorithm has some advantages and limitations. The performance that each algorithm provides varies basing on the geometry of the path and the obstacle distributions. Bug2 algorithm shows a faster performance than Bug1 when the robot navigates in open and wide paths, since it doesn’t need to implement a complete turnaround the obstacle as in Bug1, rather it will leave the obstacle soon when it hits the leaving line. On the other hand, Bug1 algorithm shows faster performance when the robot has to move in Maze paths, since the Bug2 will circulate several times till

it reaches the goal as it could be seen in fig 2.12-c. Moreover, the Bug1 algorithm is more conservative than Bug2.

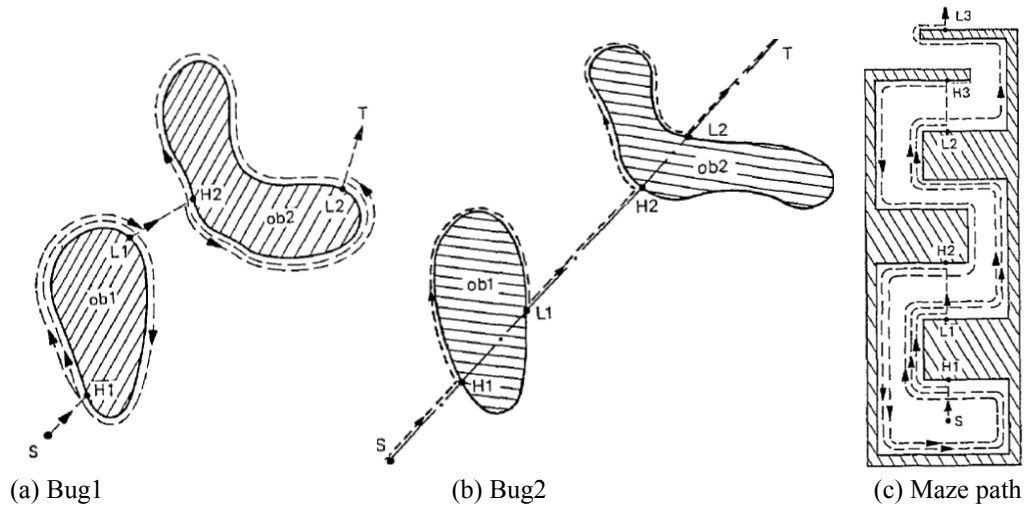


Figure 2.12: The motion of the robot for the Bug1, Bug2, and Maze path [80]

In methods, which are related to the robot's dynamics, the robot specifies the location of the obstacles, then it implements the obstacle avoidance path basing on the robot's dynamics as velocity and acceleration. Dynamic Windows Approach (DWA) is a well-known algorithm in obstacle avoidance [81]. In this method, the robot identifies the location of all obstacles in its path; then it will calculate the whole admissible linear and angular velocities. The admissible velocities are the velocities that the robot can use with the ability to decelerate and stop before reaching to the closest obstacle. The robot will then consider only the velocities that the robot can reach given the acceleration limitations of the robot. Fig 2.13 shows how the robot specifies the admissible velocities for a given work area; the grey area

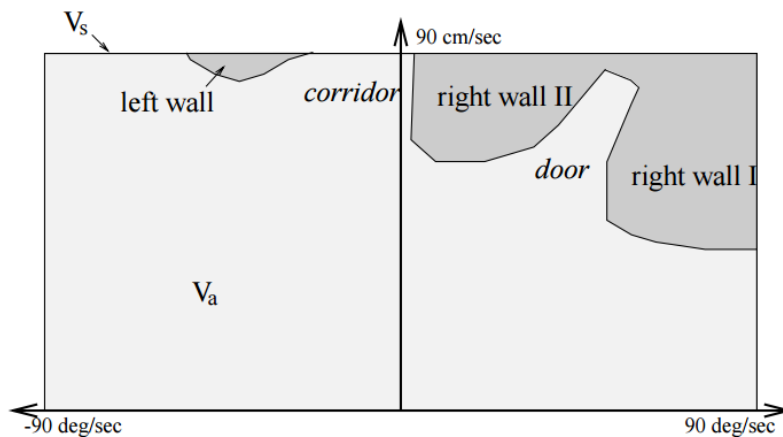


Figure 2.13: The representation of velocity space in DWA [81]

represents the non-admissible velocities that the robot mustn't use. Finally, the robot optimizes its velocities by taking into consideration the target direction, and the distance of each obstacle from the robot [82]. The experiments show that the robot is able correctly to avoid the obstacles and move in narrow corridors without oscillations. Furthermore, tuning the parameters of optimization equation plays a major rule in the motion of the robot.

**Claes et al.** presented a new method for multi-robots collision avoidance [83]. Each robot should calculate its position and velocity, and then a communication network is used to exchange the data of position and velocity. The robots will calculate the admissible velocities that each of them can use without collision. The research focuses on solving the corridor problem, when two robots are moving in the same corridor. Since each robot is moving, the collision avoidance system should be reactive to update its motion, and to inform the other robots about its direction. Each robot exchanges the uncertainty to specify the movements, which each of them should implement to pass the corridor. The experiments were done over two Turtlebot robots in a narrow corridor (around 140cm); the diameter of each robot is 33.5cm and proved the ability of these robots of passing each other without collision [84].

Reactive collision avoidance methods are those, which are based merely on the sensory information rather than the robot's dynamics. In these methods, the robot uses the sensor's reading to update its orientation and avoid the obstacles. Examples of reactive methods are potential field [85], vector field histogram [86], nearness diagram [87], smooth nearness diagram, and follow the gap [88].

Potential field local path planning which is developed by **O. Khatib** in 1985 is one of the early real implementation related to obstacle avoidance in robotics [85]. In this method, the goal location and the obstacles practice attractive and repulsive fields over the robot. The goal location tries to attract the robot to it, while each obstacle tries to push the robot away from it. The combination of all fields which effect the robot, will lead the robot to move away from the obstacles, and move toward the goal location. Unfortunately, this method is suffering from local minima problem in which the robot will get stuck when the robot got into U shaped locations, besides to the oscillating motion of the robot in narrow corridors. Fig 2.14 shows the concept of the potential field.

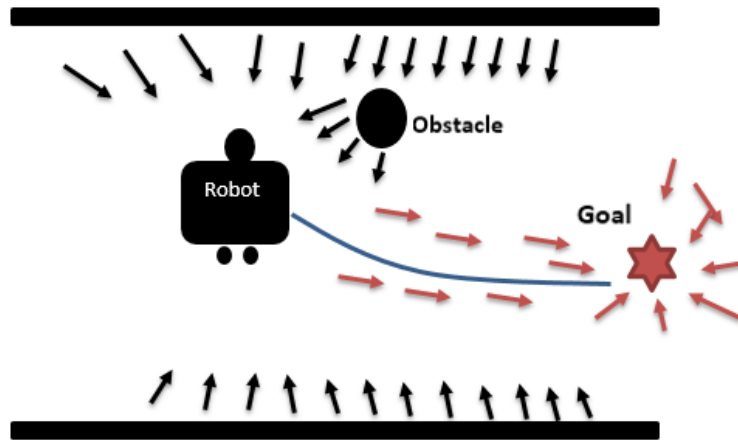


Figure 2.14: The general concept of potential field method

In 1991, **Borenstein et al.** developed a new method for collision avoidance by representing the obstacles in a histogram [86]. This method is called Vector field histogram (VFH). In VFH, the robot divides the sensed area into sectors; in each sector, the robot measures the distance of the obstacle inside this sector. In a next step, the sectors are plotted as a histogram which each bar of it describes the distance between the robot and the obstacle existed in the specified sector. The 2D histogram is mapped then to a one-dimensional polar histogram around the robot as it could be seen in Fig 2.15. A candidate valley is a group of sectors that are less than a certain threshold which represent a possible free area that the robot can navigate in it. The robot will finally select the valley that is closest to the goal location. The experiment is done by using a mobile robot with a diameter 0.8m, and several obstacles are distributed in the work area with a spacing around 1.4m. The

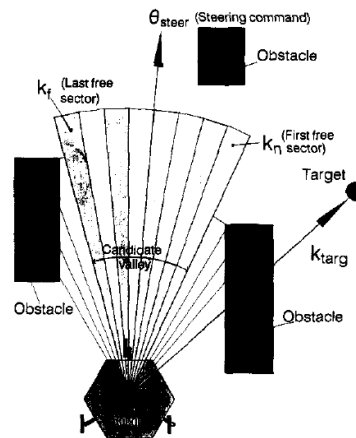


Figure 2.15: The polar representation of the VFH and the direction selection [86]

robot used ultrasonic sensors to detect the obstacles, and the velocity is set to 0.58m/s. The robot could successfully create the polar histogram for obstacles' distribution after each measurement update, and it reached to the goal without collisions. On the other hand, the method is still suffering from dead-end situations (as U shaped obstacle) which lead the robot to move around in circular paths without the ability to implement a path out of the trap. Another improvements were added to the method such as VFH+ [89] and VFH\* [90].

Nearness Diagram is a reactive collision avoidance method which is developed by **Minguez** in 2004 and follows the divide and conquer strategy in implementing the local path planning [87]. In this method, the work area is represented in sectors; each sector represents the nearness of the obstacle from the robot within this sector as it be seen in Fig 2.16. The robot defines the gaps which represents a border line between two adjacent sectors that the absolute difference between them is greater than the robot's diameter. Two adjacent sectors represent a region. Basing on the region distributions, nearness of obstacles to the robot, and the wideness of the regions, the robot will select one of five situations which to calculate the angular and linear velocities which allow the robot to avoid collision with the obstacles. The experiments are done over a mobile robot with a diameter 0.48m, which is equipped with a laser rangefinder sensor, and a maximum velocity 0.3 m/s. The robot could avoid obstacles keeping a distance 10cm from it. The time required by each path is related to the density of obstacles and their nearness to each other, since the robot will adjust its velocity based on the situations that it will face during its navigation to the goal. Furthermore, the robot didn't face the deadlock problems when it avoided U shaped obstacles, and it doesn't have local minima problems. An improved version for the ND was implemented which added additional situation to the previous ND method, and called ND+ [91].

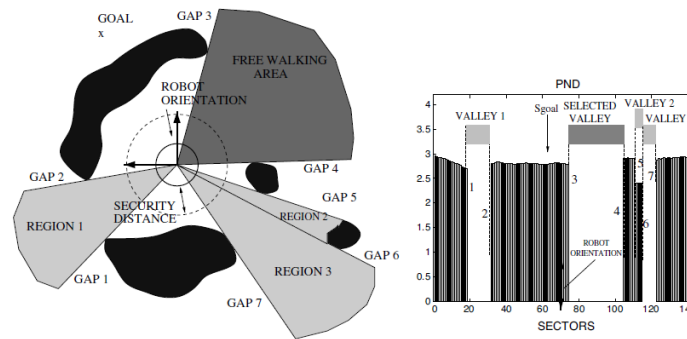


Figure 2.16: The representation of obstacles, gaps, regions, and valley in nearness diagram [87]

Smooth nearness diagram (SND) is an improved version of the ND+ method, and developed by **J. Durham** in 2008 [92]. The main goal of the improvement is to simplify the ND method by using single law which is valid for the whole scenarios that the robot faces instead of using six scenarios. After detecting the regions, and selecting the valley, the method defines three angles: the safe rising gap  $\theta_{srg}$ , which is the calculated angle from the selected gap; the  $\theta_{middle}$  which is the angle that directs the robot to the middle of the selected valley, which is important to access the narrow valleys, and the desiring heading angle  $\theta_d$  which is equal to  $\theta_{srg}$  or  $\theta_{middle}$  whichever the closest angle to the angle of the rising gap. Fig 2.17 shows the concept of the SND. The experimental results show that the SND completed a given narrow path successfully within 135 seconds, while the ND+ crashed with the last obstacle in the same path and it required 254 seconds to finish it. Furthermore, the experiments show a smoother path by SND than ND+, due to the transitions between situations implemented by ND+. In [93], it is shown that the SND suffers from the deadlock problem which occurs since the method calculates the total weighted deflection despite of the obstacles' distribution. Thus, when the robot moves in narrow path and more obstacles in one side than the other side, the calculation of deflection will lead the robot to move far away from the side with higher risk, causing the robot to collide with the obstacle with lower risk.

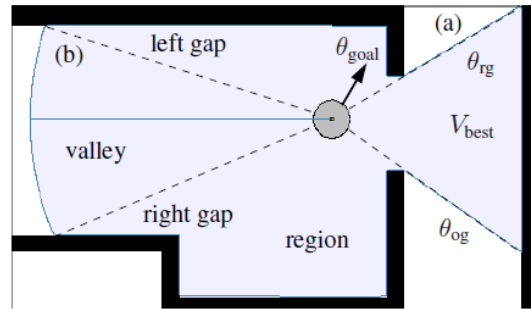


Figure 2.17: The concept of implementing collision-free path using SND method [92]

In 2010, **Mujahad et al.** presented a new collision avoidance system based on the ND and SND methods called “Closest gap”(CG) [93]. In this method, instead of evaluating the whole gaps, the robot will ignore the gaps that are included into other gaps. Furthermore, the method tries to overcome deadlock problem which is faced in SND, by distinguishing between the obstacles to the right and left sides of the robot, and calculate the deflection issued by the obstacles located at each side and then calculate the total net deflection resulted from the right and left deflections. The experimental results show that the proposed method has smoother trajectory than the ND and SND which caused by the motion oscillation of the robot when it tries to plan its path to the goal, besides to avoiding the deadlock problems. Furthermore, a



simulation is done to compare the time required by SND and CG, and it shows that the SND could reach the goal in 140s, while the CG reach the goal with only 125s.

**Kim et al.** presented a collision avoidance method based on extracting the borders of the path, and the location of obstacles in the path [94]. The robot uses CCD camera to extract the boundaries of the lane that it navigates through. These boundaries meet in a point called vanishing point as it could be seen in Fig 2.18. Extraction of the continuous lines is implemented using Hough transform; while boundaries recognition is implemented by RANSAC algorithm. The robot uses the vanishing point to guide itself through the path. When the robot meets an obstacle, it will calculate the steering angle taking into consideration the angle of the obstacle, and the angle of the vanishing point. When the robot avoids the obstacles, it will direct itself to meet the vanishing point and resume its path. The experimental results show that the method has a better performance than DWA and VFH in case of travelling time, and travelling distance.



Figure 2.18: The vanishing point-based navigation [94]

**Hagiware et al.** presented the use of Kinect sensor in navigation and obstacle avoidance for the indoor mobile robots [95]. The robot records the path that it will follow, and then it will detect its position and avoid obstacles by comparing the captured frame with the recorded frames in its memory using three steps. In the first step, the robot implements view matching by searching for the most similar recorded frame to the current frame using SURF algorithm. In the second step, the robot uses ego-motion algorithm to for estimating its relative linear and angular distances between the recorded frame and the captured one, and the rotational and positional vector of the robot is then extracted. When an obstacle is detected in the path, the robot will calculate its position and the position of the obstacle by comparing its current frame to the stored frames, and implement a diagonal collision-avoidance path around the obstacle. The experimental results show that the robot is able to estimate its location even when it strays from its recorded path, with the ability to avoid the obstacles that it faces in its path. [96 - 105].

To summarize the obstacle avoidance algorithms, table 2.7 shows the advantages and limitations of the most common methods in obstacle avoidance for mobile robotics; while table 2.8 shows the common sensors which are used to detect the obstacles located in the path of the robot.

Table 2.7: Comparison between collision avoidance systems

	Advantages	Limitations
Bug1	Doesn't suffer from local minima problems Reliable method to reach the goal location	Turning around each obstacle causes the method to be slow [106]
Bug2	Doesn't suffer from local minima problems Faster than Bug1 in most situations.	The method shows low performance in maze paths. [106]
PF	The integration of the whole obstacles together with the goal location to generate the collision-free path	Trap situation "local minima" [107] Oscillations in narrow corridors [107]
VFH	Insensitive to misreading of the sensors [86] Fast computation with fast motion [86]	Difficult to tune threshold "a threshold which is good for cluttered path, not good for obstacle-free path" [87] Low performance with close obstacles [93]
DWA	Takes into consideration the robot's physical limits (velocities, accelerations,...) [81] Experimental results show good performance for the method.	Difficult to tune the parameters Problems in avoiding the U-shaped obstacles [87]
ND/ND+	Doesn't require tuning (one parameter in one situation) [87] The method shows good performance in moving in cluttered and dense environments [87]	Not suitable for noncircular robots [87] Changing between situations limits the smoothness of the path (sharp transitions) [92]
SND	Smoother paths compared to ND Single motion law instead of six in ND [92]	When the robot faces more obstacles in one side than other, the method will guide the robot far from the side with more obstacles causing it to collide with obstacles in the other side [93]

CG	Reduced path calculation, faster, with less oscillations compared to ND and SND [93] Remove the gaps that are contained in other gaps. Overcome the deadlock problem.	Since it is relatively new method, no prove for a limitation.
VP	Higher travelling velocity compared to VFH and DWA [94] Combine line following with obstacle avoidance to generate flexible obstacle avoidance paths.	Implemented over straight paths, without tests for the curved paths with cluttered obstacles. Works only with vision-based sensors

Table 2.8: Comparison between different kinds of collision-avoidance sensors

	Advantages	Disadvantages
Ultrasonic	Linear Performance Easy to program Low current consumption	Can't detect human body Effectuated by voice noises
PIR	Human detection Low power consumption	Doesn't provide rich information about the body location and distance. Passive sensor. Effectuated by ambient light.
Vision sensors (Eye cameras for the H20 robot)	Low cost Contains many data about the environment	Difficult to program Requires high computations
Kinect	Could be used as vision sensor Gesture Recognition 3D vision (with depth) Could be used for human-robot interaction and obstacle detection.	Effect of ambient lights High power consumption Depth camera can't detect obstacles located < 50 cm from the robot Noises over the skeletal frames resulted from robot's vibration
LRF	Accurate sensor with robust obstacle detection. Could be used for localization and mapping.	Can't detect transparent obstacles.

In conclusion, many collision avoidance systems are proposed for mobile robots. Each of them shows advantages with some limitations under certain conditions.

Moreover, different sensors are used in detecting the obstacles in the work area. It could be seen that most of the collision avoidance methods gave much more focuses on the static obstacles, with minor experiments over the dynamic obstacles, especially the human which are considered as “intelligent” dynamic obstacle which will also try to escape the robot by generating collision-free path. This is a short come in the majority of the obstacle avoidance methods. Thus, in this work new concepts will be proposed for the task of collision avoidance in the existence of human in the same work area. The concepts will consider that both the human and the robot have to incorporate in generating the collision-free paths, so they can avoid each other safely.

Since the proposed collision avoidance system will be based on the interaction between the robot and the human, the Kinect 2.0 sensor will be used in this work for detecting the human and get the gestures from them. The Kinect 2.0 is a 3D sensor, which has two cameras (RGB and Infrared) enabling the robot from identifying the objects and human’s skeletons, which are existed in the area. Skeleton frame provides the skeletal information of up to six human whom are located in the vision area of the sensor. The detailed information regarding the proposed system together with the implementation and experiments are shown in chapter 6.

## Chapter 3

# Goals of the Dissertation and Realization Concepts

---

### 3.1 Background and Work Description

The new technologies which appeared in the last few decades, changed the techniques and work methods in different work sectors, and boosted the production speed and quality. Laboratory is one of the work places which affected by these new technologies [6]. Engineering changed drastically the fundamentals of preparing and handling samples, with decreasing the processing time, and costs. This includes sample processing and sample management [108], sample handling [8], laboratory management [109], and robotics [110].

For any industrial product, the materials have to be processed and handled by different machines, and transferred between these machines as fast as possible to decrease the processing time and increase efficiency. Actually, conveyor belts and robotic arms are widely used nowadays in manipulating the products, in chemical laboratories as example. On the other hand, there is a dilemma of manipulating these samples between different laboratories. Thus, one of the important link in the integration chain is a global transportation system which is able to manipulate and transport the different samples between these laboratories at any time. Such system must be flexible so it can be easily modified to meet the adjustments in work area as adding/removing laboratories or adjusting the pick and place locations. Furthermore, the system has to be secure so it doesn't cause accidents or failure during the manipulation tasks.

Fortunately, the fast achievements in the field of indoor mobile robotics bring a promising solution for bridging the gap of mobile transportation of samples and lab ware between the laboratories without the need for human help.

In mobile robotics, several tasks have to run in parallel to guarantee a smooth and safe operation for the robots during implementing the transportation tasks between the laboratories. A global map for the work area is required for any mobile robot, to enable the robot to define the goal location and the locations of other waypoints as charging or grasping locations [12]. Furthermore, the robot must be able to localize

itself robustly within the global map despite of the noises which may affect the sensor's measurements [111]. The robots will work side by side to human; they must be able to recognize the human, interact with them, and obey the orders given to the robot via interaction [112] [113]. Moreover, the robot must be able to avoid the collision with the human located along its path using local path planning strategies [12]. Also, the robot has to define the objects that it has to transport, grasp them using its arms, and place them in the right locations [13]. Finally, the robot must be able to work 24 hours, which means that they have to be able to reach the charging stations and charge themselves autonomously [114]. Fig 3.1 shows the sub-systems which are working in H20 robots, and which allow the robot to grasp objects, transport them safely between different floors, interact with humans, avoid collisions by modifying its path temporally, besides to communicate with the process management system to receive the transportation requests and send reports to the process management system regarding the robot situation.

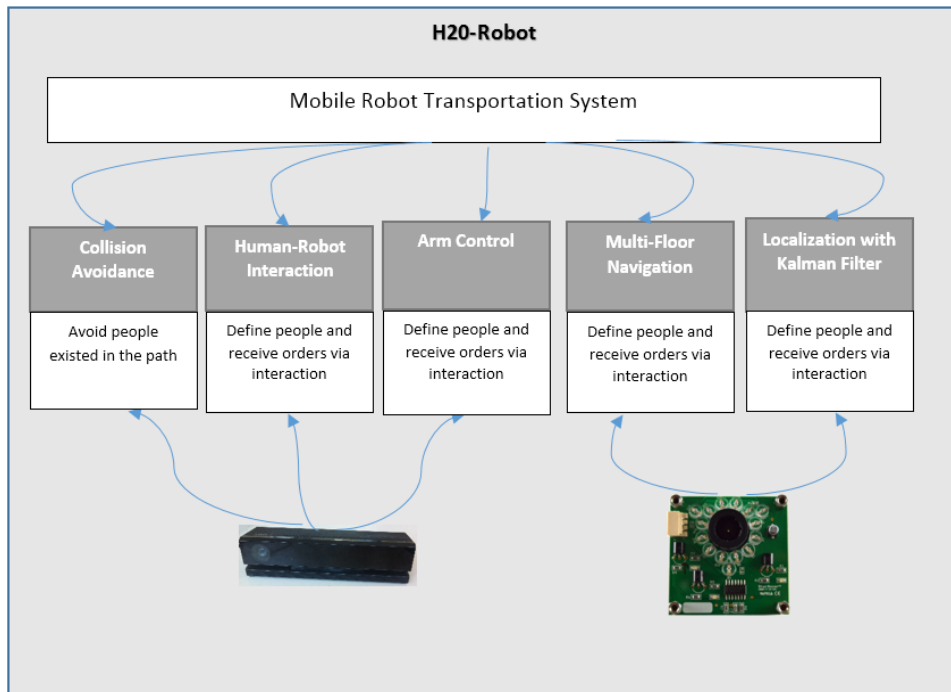


Figure 3.1: The integration of robotic tasks into the H20 robot

For achieving the research goal (secure navigation), it should be taken into consideration implementing a robust localization and mapping system which is able to accurately localize the robot under different work conditions. Furthermore, it should be taken into consideration that the work environment includes robots which are working side by side with humans. Thus, safety is a major challenge to make the manipulation system applicable.

From the previous discussion, it will be clear that any implemented system won't be applicable without being integrated with the other systems. Only this integration will guarantee a successful transportation system based on mobile robotics. For instance, the robot will move to the goal location based on the global map as long as no obstacles are existing in the work area. When a person appears, the collision avoidance and the human robot interaction systems must be activated and control the robot and implement a local path to avoid the collision with the human, and return the control to the global navigation system when no human are existing in the path.

The aim of this research is to improve the safety of the mobile robots which are working in social environments, by finding robust sensors and developing algorithms to enable the robots to identify their location, recognizing the surrounding environment and interact with the changes in the environment by modifying the behaviour in response to these changes. This work focuses on three aspects, which are necessary for a safe navigation for indoor mobile robots, which are localization, human-robot interaction, and collision avoidance. Fig 3.2 shows the H20 humanoid robot, which is used in this dissertation for testing and improvements.

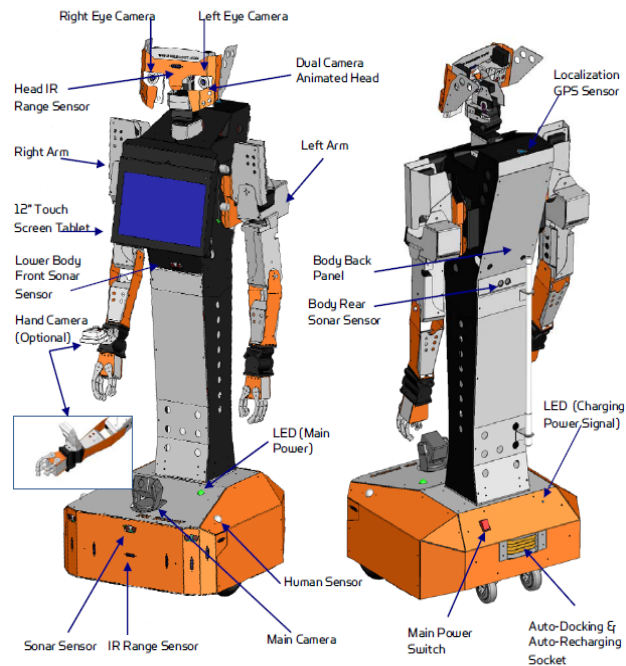


Figure 3.2: The structure of H20 Mobile Robot [115]

The H20 humanoid mobile robot has dual arms with 6 degree of freedom (DOF); a moving head with 6DOF and two RGB cameras. The maximum speed of the robot is 0.75 cm/sec. Moreover, the robot is equipped with a GPS localisation sensor based

on IR technology, 5 sonar sensors for collision avoidance tasks, 10 IR range sensors, and two passive infrared sensors PIR. Furthermore, the robot is equipped with a powerful on-board computer, with Kinect 2.0 sensor, and additional batteries to power the Kinect sensors.

### **3.2 Improving the localization of indoor mobile Robots**

For any mobile robotic system, robots require a map of the work environment, and they need to identify their location in real time within this map. Any error in the localization system could lead the robot to lose its location, and move in wrong directions, which might lead to accidents. It could be seen from the previous literature survey, that many localization methods are available for the mobile robotics, and each has its limitations and advantages. Furthermore, most of these methods require algorithms and further processing for the sensory data. A good selection for the localization system depends on the application and the work environment.

In general, the localization sensors can be divided into relative sensors and absolute sensors. In relative sensors, the updated measurement will be based on previous ones. Thus, the error in any step will affect the following measurements, and this will lead to an accumulation of the errors, and the measurements will be totally false after a time. Examples of relative sensors are encoders, inertia sensors, and dead-rocking sensors. In absolute sensors, the sensors don't have accumulated errors; they provide measurements basing on the current state, without the need for previous knowledge of the location. On the other hand, the error in this kind of sensors will be instantaneous, and could cause accidents when the robot fails in correctly detect its location. In this work, StarGazer sensor from Hagisonic company (South Korea) is selected for the localization and mapping [45]. This sensor is based on infrared technology, and it is belonging to absolute sensory family. The sensor is composed of an electronic chip which has an infrared camera and a group of infrared emitters, and passive landmarks which are distributed over the work area. The sensor works on analysing the received infrared beam from passive landmarks to obtain its location. Unfortunately, this sensor is affected by light noises resulting from sunlight and florescent lights. When the robot moves in noisy areas, the sensor will falsely define its location in the global map. Fig 3.3 shows an example of a scenario for false detection of a landmark for a certain period of time. The robot will start moving from location 1 to location 2, and then it will move to location 3. Supposing that the location 3 includes light noises, the robot will fail to detect the correct location of the robot, and it will suppose that it is in location 6 as example. Consequently, this will lead the robot to adjust its path to direct itself to the next waypoint following location 6, while it is still in location 3.



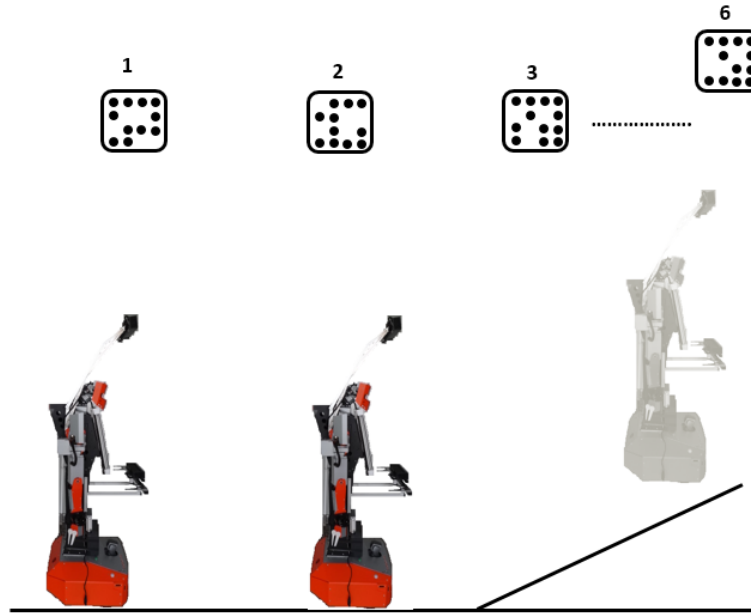


Figure 3.3: False detection of landmark

To overcome this problem, the measurements of the StarGazer sensor must be filtered to check the false measurements, and provide estimation for the location of the robot under these noisy areas, to allow the robot from keeping its movement to the goal location. In the literature survey, it is shown that the false detection doesn't always follow white Gaussian noise, so implementing recursive filters such as Kalman filter won't provide accurate estimation for the robot's location. Thus, a modified Kalman filter is applied to the measurements of the StarGazer sensor. In this work, the adjusted filter will monitor the false measurements which are far from the expected location of the robot, and remove it before implementing the Kalman filter over the measurement. This filter uses the history measurements to predict the location of the robot. When a new measurement is far from the predicted one, the filter will know that this value is wrong, and the robot will use the predicted measurement to keep moving in its path till the robot moves away from the noisy location.

### 3.3 Human-Robot Interaction

Human-Robot Interaction (HRI) can be defined as the ability of a robot to recognize the human, interact with them, and implement suitable activities as a response for the interaction. The task of HRI varies; it could be used to teach the robot how to implement certain tasks as it could be found widely in cognitive robotics; others employ the HRI to help the human in their daily life as it could be seen in human-

assistive devices/robots, and in robotic arms which work with human in assembly lines. Social robots also use the HRI to express the emotions either via voice sentences or via physical behaviors as moving the head, eyebrows, and mouth; and the interaction which aims to control the motion of the robots and its behaviour.

For any engineering system, safety has the highest priority which the system has to achieve. Furthermore, these systems must be equipped with facilities that enable the human from interrupting the system and control it when needed. Such as stopping the system, or controlling it in emergency situations.

In mobile robotics, robots will navigate in social environments, and they have to guarantee the safety of the human who are working in the same area, by recognizing the humans, interacting with them when needed and avoiding physical accidents to them. Moreover, and as an engineering system, the robots must be controllable, and interruptible by human when needed.

In the literature survey, there is no research took into consideration these concepts; researches focused either on implementing collision avoidance systems which allow the robot from navigating in narrow, cluttered areas with distributed obstacles, or on implementing human-robot interaction systems that don't serve the task of secure operation.

In chapter 2, it is shown that the collision avoidance serves as a local path planner; it allows the robot from adjusting its path temporary to avoid the collision with the obstacles that are located in the detection range of the sensor. The majority of collision avoidance systems handle the human as a dynamic obstacle, without taking into consideration the reality that the human "as an intelligent moving object" will also try to adjust its movement to avoid the robot. Fig 3.4 shows a scenario when the robot and human meet in a certain location. It could be seen that the robot will adjust its path trying to avoid the human on its right/left. Simultaneously, the human will also try to avoid the robot, so he will adjust its path to avoid the robot taking the left/right direction. This will make both the robot and the human confused about the motion direction that the other will follow.

In some scenarios, the human might have better information regarding the obstacles that are located in the path, and if he can interact with the robot, he might give it a better direction for implementing the collision-free path. As example is when a group of human are located in the path of the robot; then with the previous collision avoidance systems, the robot will try to avoid the human autonomously, and this will confuse the human who are located near the robot. If the human is able to interact with the robot, and provides the direction of the collision avoidance to it,

then the robot and the human will know the behavior of the other and the direction that each of them will follow.

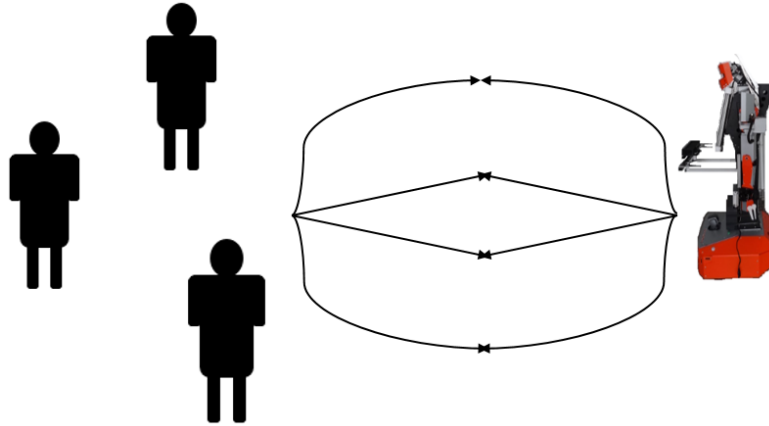


Figure 3.4: The conflict in generating collision-free path between the robot and the human

In bottleneck problem which is described in [104], it is likely that the robot and the human meet in a narrow corridor or in small locations so it is not possible for any of them to keep moving. Most researches proposed systems for enabling the robot to recalculate its path and adjust it autonomously with the goal of avoiding obstacles. Consequently, when the robot meets human in a narrow area which is not wide enough for avoiding each other, the robot will either stop, or keep moving and collide with the human. Thus, the HRI will allow the human to interact with the robot, and adjust its motion temporarily. The user can ask the robot to move forward or backward for a certain distance, till the robot reaches a wide location which is enough for the human to avoid the robot and keep moving.

In this work, the HRI will be employed to serve the task of secure operation of the indoor mobile robots to guarantee the safety of the human and the robot, by providing the facility for the human to interrupt the motion of the H20 humanoid robot, and modify its motion in certain situations as guiding it to a certain location, or to cooperate in avoiding collision when both of them located in narrow corridors or in cluttered environments.

In this method, both the human and the robot will share the responsibility of avoiding each other via interaction. The robot will execute the motion orders issued by the human via interaction. When the robot meets human in its path, it will receive the orientation, which the robot must go to avoid the human. This will guarantee that both the human and the robot know to which direction they have to go to avoid

collision. Furthermore, it is taken into consideration the distraction of human while he gives orders to the robot. This is done by limiting the interaction to a single person, and reading several gestures from the human for a certain period of time, and executing the order when the whole gestures which are taken within this period are identical.

From the literature survey, the HRI could be classified into three categories:

- The physical HRI: the robot obtains the interaction information from the human using contact sensors which are in contact to human as wearable sensors, and touch sensors.
- The non-contact HRI: the robot gets the interaction information from the human without contact as gesture and voice recognition methods, using microphones, RGB, 3D and LRF sensors.
- A combination of the two previous methods: as fusing the sensory information of LRF and wearable sensor.

Since the interaction will serve the collision avoidance and human safety, it is required to use a sensor which is able to detect the human and interact with them remotely. Thus, the Kinect sensor is chosen for this work. This sensor provides skeleton frames to the robot with a 3D description for the joints' positions, without describing the activities that a person is doing. Thus, the robot has to process the joints' orientations of each person in the skeletal frames provided by Kinect sensor to extract and define the activities that the human is doing and implement the correct response for these activities.

In literature survey, many methods were used to classify the gestures using the Kinect sensor; each of these methods has its limitations and advantages. In methods which depend on analyzing the relative positions of the joints, and the geometry analysis of the joints' angles will provide fast recognition of the gesture. On the other hand, such methods are sensitive to the angular displacement of the human; when the human and the robot are not located face to face, the angles between the joints will be distorted, and the HRI system will fail in detecting the gesture. Thus, in this thesis, machine learning and artificial intelligence algorithms were adopted to train the models to detect the gestures of human even if they are not located face to face with the robot. Two algorithms will be used and compared to classify the gestures of the users. The first algorithm is Support Vector Machine (SVM), and the second algorithm is Back Propagation Neural Network (BPNN). These two models will be trained and employed to classify the joints' coordinates of each person detected by the Kinect sensor. Then the performance will be compared to select the suitable one for classifying the gestures.

SVM is a machine learning method, which is based on statistical learning theory (Vapnik 1995) [116], [117], [118]. Briefly, the SVM principle is based on classifying the data into classes using separation planes. When the data is not linearly separable, the SVM maps it into a higher dimensional space using kernel functions as Gaussian, radial basis function, and Polynomial kernels, and then the model searches for the optimal hyperplanes which are able to separate the mapped data.

Back Propagation Neural Network is a supervised learning algorithm which is used in several applications as in pattern classification, pattern recognition, and image processing. The network is composed of three layers: Input layer, hidden layer and output layer [119]. Each layer includes several neurons. Each neuron existed in a certain layer is interconnected with each neuron in the following layer with a weighted connection. Furthermore, each of the neurons in the hidden and output layer has an activation function which stimulates the neuron to provide the output. To train the network, a set of training data is used to tune the weights of each connection. The training samples are applied to the input neurons, and the output is compared with the expected output to calculate the error. This error is propagated backward to update the weights of each connection to decrease the model error.

To use SVM and BPNN in real applications, the models have to be trained before using them in classifying the gestures of the users obtained by the Kinect. Cross validation techniques are statistical learning algorithms which split the available training data into two sets: training set, and testing set. The training set is used to train the model, while the testing set is used to validate and analyze the trained model, to check how its performance before putting it into real application. The K-fold cross validation method will be used in this work to train the model. In this method, the training set is divided into equal subsets. At each training cycle, a subset is selected as testing set, while the remaining data will be used to train the model. Further details will be described in chapter 5.

After extracting the gestures and classifying them by the HRI system, the robot has to implement suitable responses for each order issued by the user via interaction. In this thesis, the interaction aims to guarantee the safety of the human, and avoid the collision and bottleneck problems. The orders “move forward/backward” will allow the robot to move toward/against the position of the user; the orders “move right/left” will order the robot to search for the collision avoidance paths to the right/left of the user; “stop” will make the robot keep stopped; and “master select” which will be used to limit the interaction to one person in a group. Further details can be found in chapter 5.

### 3.4 Collision Avoidance for Indoor Mobile Robots

Collision avoidance is a basic requirement for any robotic system, to guarantee the safety of the robot and the human from physical accidents which could occur when both of them meet in the same path.

Different collision avoidance systems have been implemented for mobile robotics. Many of these systems proved their validity in avoiding static and dynamic obstacles, and navigating in cluttered environments. Nevertheless, most of them handle the human as dynamic obstacles, rather than as “intelligent obstacle”. This is a short come since the human will also try to find a collision-free path, and this could cause a confusion due to the lack of knowledge regarding the direction that the other will take.

Out of this, a two-level collision avoidance system has to be implemented, which takes into consideration the human obstacle as an “intelligent obstacle”. The system gives a common responsibility for both the robot and the human in avoiding each other. The method has two levels:

- **Cooperative Collision Avoidance:** When the robot meets a person in the path, it will ask the human via voice messages for interaction to supervise the collision avoidance process. The user in this case will be delegated by the robot to select what it has to do. The user will then be able to order the robot to either move forwards/backwards when they are located in narrow paths, or to move right/left so the robot generates collision-avoidance path to the right/left of the user. In this case, both the robot and the human know the motion of the other, and this avoids a conflict when each one moves independently from the other.
- **Autonomous Collision Avoidance:** If no human interacted with the robot for a certain period of time, the robot will calculate the collision avoidance path autonomously, taking into consideration finding the closest collision-free path to the goal location.

To realize these concepts, a robust collision avoidance system is implemented. In this system, the robot will search for the whole navigable regions that are wide enough to allow the robot from passing the human safely, taking into consideration the width of human and the robot. Furthermore, the system will provide two collision avoidance options: autonomous and cooperative.

The robot will stop and ask the human to interact with it; allowing a time period 3000 ms for the human to implement the interaction. If a user raises the right arm vertically ( $180^\circ$ ) for a period around 400ms, the robot will know that it has to interact with this user and execute his requests provided via the interaction, and the

robot will switch to cooperative collision avoidance. Thus the selection of the collision-free path will be based on the direction provided by the master.

If no human interacted with the robot within the given period of time, the robot will switch to autonomous collision avoidance. Thus, it will select the region that it will generate the collision avoidance path across it basing on the next way-point that the robot has to move based on the global map.

Finally, the collision avoidance system is equipped with a velocity controller which adjusts the robot's linear and angular velocities based on the width of the region that it will pass. The controller will decrease the robot's velocities when the selected region is narrow, and vice versa, and this will promote the performance of the collision avoidance system.

## Chapter 4

### Localization of Indoor Mobile Robots

---

#### 4.1 Introduction

Localization is defined as the ability of the robot to estimate its position given a map of the environment [120]. For successful indoor robot navigation, the robot requires a map building for the workspace, and sensors for specifying its position within this workspace. If a robot loses its position within a work environment, then it is probable that fatal accidents occur, like colliding with walls or doors. In life science laboratories, accurate localization is critical, due to narrow corridors and small free spaces, which don't enable robots to have wide localization tolerances.

Furthermore, accurate localization is important to make sure that robots are able to reach pick-and-place stations, elevators, and charging places. The accurate localization will be based mainly on selecting suitable sensors, which meet the requirements of the work area, as specifying the possible disturbances, which may exist in the place. There is a wide option for sensors, each has its benefits and limitations, and a good selection for sensors must take into consideration the application requirements.

For example, in life science laboratories, it should be taken into consideration that these laboratories are subject to electromagnetic noise, ambient light, the existence of narrow corridor and small free spaces, besides to the need to modify the navigation map due to a change in the work environment etc. Based on the previous survey, the StarGazer sensor (Hagisonic, South Korea) is selected for the tasks of localization and mapping [45]. Fig 4.1 shows the principle of the sensor.

The sensor is equipped with an array of IR emitters, and an IR camera positioned in one circuit. Furthermore, a collection of the passive Landmarks is distributed along the navigation area in celisca. The passive landmarks are composed of 4x4 points, which have a high reflective ability for IR beam. Each landmark has its unique ID based on the points' distribution over the landmark. Figure 4.2 shows the StarGazer sensor and the landmarks. The StarGazer emits the IR beam to the landmark, which in turn reflects the IR beam back to the StarGazer. The IR camera and the integrated circuit analyses the camera's data to obtain the position and orientation of the robot.



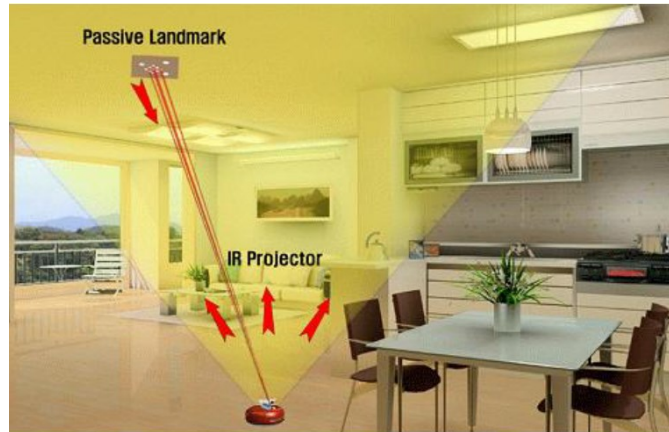


Figure 4.1: The principle of StarGazer Localization sensor [45]

From Fig 4.2, it could be seen that the passive landmarks are composed of several small circles; each of these circles is composed of a reflective material which is able to reflect the IR beam effectively. The StarGazer will then identify the location, direction and the landmark ID by analyzing the reflected beam from the circles. The StarGazer is able to recognize up to 4095 different landmarks for the landmarks which are composed of 4x4 dots. Further information of the sensor could be found in appendix 1.

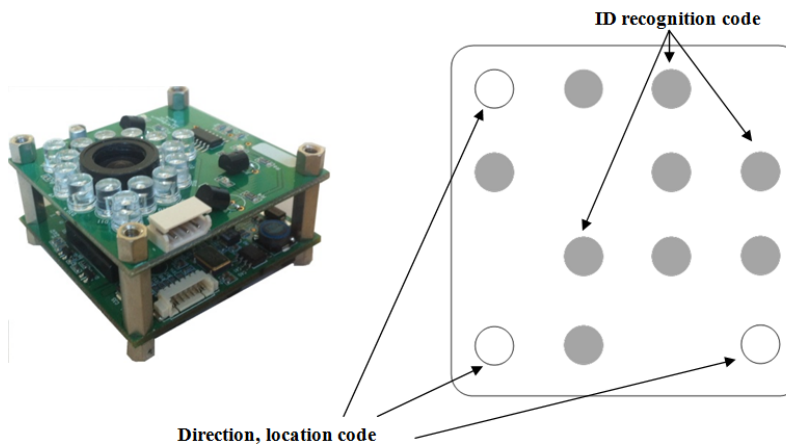


Figure 4.2: StarGazer sensor and the passive Landmarks

## 4.2 Improving the StarGazer Localization using Kalman Filter

As it is mentioned in chapter 3, it is noticed that the StarGazer is effected from the light noises which are resulted from strong sunlight, and from the fluorescent lights when they are located near the landmarks of the StarGazer sensor. Fig 4.3 shows an

example of a noisy path that includes lights which are located near the landmarks, and path that is subject to sunlight. When the robot moves in such path, the false detection of the sensor could occur and this will lead the robot to lose its location as it is explained before in chapter 3, and Fig 3.3.

To improve the performance of the StarGazer sensor and filter the false measurements, Kalman filter is employed for removing the false measurements and providing estimation for the robot's location under wrong measurements.

Kalman filter is a recursive data processing algorithm; which is able to incorporate all provided information. The filter is able to process all the available measurements, to estimate the current value of the variables of interest. Furthermore, the filter is able to predict the state of the system based on previous measurements.



Figure 4.3: The navigation under strong natural and fluorescent lights

The filter has been used in several applications such as rocket navigation [121], object tracking [122], wind energy [123], [124], power systems [125], speech enhancement [126] etc.

To implement the Kalman Filter, it is required to model the system dynamics, and to know the initial state of the system. The filter is composed of a set of mathematical equations [127]:

$$\hat{X}_{k+1}^- = A_k \hat{X}_k + B_k u_k + w_k \quad (4.1)$$

$$P_{k+1}^- = A_k P_k A_k^T + G_k Q_k G_k^T \quad (4.2)$$

$$Z_k = H_k X_k + v_k \quad (4.3)$$

$$K_{k+1} = P_{k+1}^- C_{k+1}^T (C_{k+1} P_{k+1}^- C_{k+1}^T + R_{k+1})^{-1} \quad (4.4)$$

$$\hat{X}_{k+1} = \hat{X}_{k+1}^- + K_{k+1} (Z_{k+1} - C_{k+1} \hat{X}_{k+1}^-) \quad (4.5)$$

$$P_{k+1} = (1 - K_{k+1} C_{k+1}) P_{k+1}^- \quad (4.6)$$

Where:

$$w_k \sim (0, Q_k)$$

$$v_k \sim (0, R_k)$$

represent the observation and measurement noises, which are assumed to be zero, mean Gaussian white noise.

$\hat{X}_{k+1}^-$	State Prediction	$H_k$	Transformation Matrix
$P_{k+1}^-$	Covariance Matrix Prediction	$u_k$	Control Vector
$Z_k$	Vector Measurement	$A_k$	State Transition
$K_{k+1}$	Kalman Gain	$P_{k+1}$	Covariance Matrix Update
$\hat{X}_{k+1}$	State Estimation Update		

The estimation step of the Kalman Filter is generated basing on the system dynamic, while the correction step will be generated basing on the measurements of the StarGazer sensor and the estimated state. In [47], it is mentioned that the noise of the StarGazer sensor comes from two resources, the noise which is resulted from the vibration of the sensor when the robot moves, and the noise which is resulted from false detection of the landmarks. Furthermore, it is shown that the first type of noise could be considered as a white Gaussian noise, while the second type of noise can't be represented as a white Gaussian noise so the recursive filters such as Kalman filter won't be suitable in filtering the measurements of the StarGazer sensor.

In the following, additional improvement to the Kalman filter is implemented, to cope the problem of the errors that don't follow the white Gaussian noise. By considering that any engineering system can run with certain limits (velocities, positions ...) that it can't exceed with a given period of time, so it would be possible to detect and delete the wrong measurements that are far from the specified limits of the system for a given time period.

### 4.3 The Improved Kalman Filter

In Kalman filter, when the filter covariance's and the gain reach the steady state, it will be able to smooth the measurements by adding the magnified difference of the actual measurement and the estimated one to the estimated measurement, as it can be seen from equation (4.5). When a false measurement is forwarded to the filter, it will update its parameters basing on this magnified difference. Consequently, if the difference is too big, it will affect the performance of the filter, and this could lead to unsatisfied results from the filter.

To implement the Kalman filter, the state estimate is calculated, and then the update equation is calculated based on equation (4.5). In most dynamic systems, it is possible to identify some certain domains for the monitored parameters when it is known that these parameters can't exceed certain limits for a given time domain. For example, in H2O robots, since it is known that the maximum robot velocity is 0.75 m/s, it is possible to judge that after 1 second, the robot won't move more than 0.75 meters in any direction (considering that the robot is moving in indoor environment with no sloping paths).

The proposed method aims to improve the performance of the Kalman filter by monitoring the absolute difference between the updated state  $\hat{X}_{k+1}$  and estimated state  $\hat{X}_{k+1}^-$  or  $|\hat{X}_{k+1} - \hat{X}_{k+1}^-|$ . When this difference is out of the known domain, then it is possible to neglect the updated state and use the predicted state as the new state of the system. Since the updated state used a false measurement. Fig 4.4 shows the flow chart for the proposed method. In this case, this adjustment over the filter will contribute in providing better results of the Kalman filter [111].

An example of the method is the H2O robot, the robot moves in a maximum speed 75 cm/sec. Thus, considering that the sampling time of the filter is 200 ms, the difference between two consequence states shouldn't exceed 15 cm. Consequently, it is possible to ignore the updated state when the difference between the predicted and updated states exceeds 15 cm. In other meaning, a measurement is considered false when the difference  $|\hat{X}_{k+1} - \hat{X}_{k+1}^-| > 15 \text{ cm}$ .

This algorithm can increase the performance of the Kalman filter, since it will monitor the updated state and reject it as long as it is out of the specified domain. This condition could be justified and tuned for other applications, which use Kalman filter for measurements filtration.

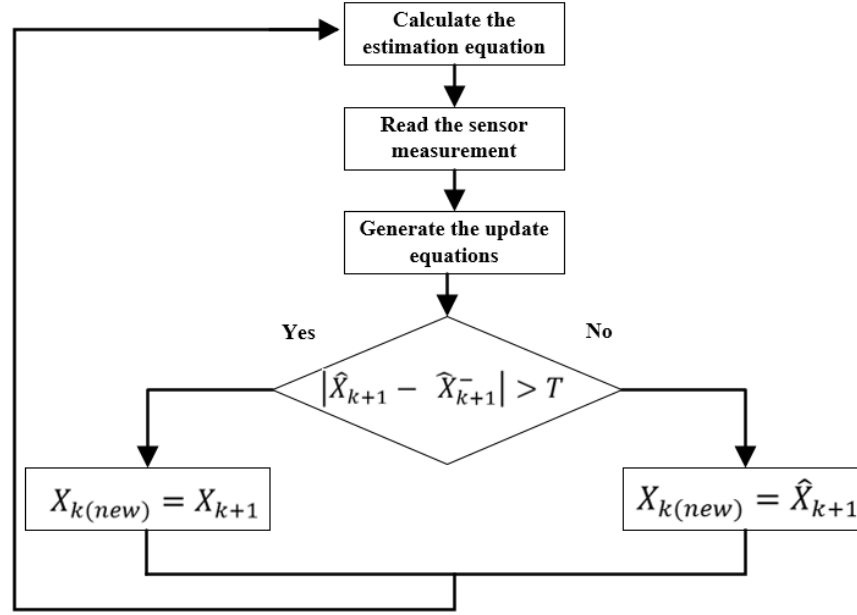


Figure 4.4: The flow chart for the proposed filter

#### 4.4 Experimental Results

The experiments were implemented with the Stargazer sensor, using 20 landmarks of type HLD1-L, which cover a path of 25 m. The experiments were done in the path which is shown in fig 4.3 and tested in different light conditions (fluorescent and sun light). In each experiment, the complete distance the StarGazer moved is 100m. The experiments have been implemented for five times in different daylight conditions.

Selecting the appropriate tolerance for the filter has a high importance for its performance, and rejecting the false measurements. Thus, each measurement is considered false when the value  $|\hat{X}_{k+1} - \hat{X}_{k+1}^-| > 15$  cm. Table 4.1 shows the experimental results. The success rate represents the performance of the filter in detecting false measurements and providing estimated ones. It could be seen that the StarGazer provides higher false measurements in strong light conditions than when the light is weak. Furthermore, it could be seen that the proposed filter proved the ability from detecting the false measurements, and providing estimation for the robot's position instead of the false values.

Table 4.1: The experimental results of the Kalman filter over the StarGazer sensor

Experiment Number	Travelled Distance (m)	Total measurements	False measurements	Light condition	Success Rate
1	100	405	14	weak sun + lights off	100%
2	100	407	21	moderate sun + lights on	100%
3	100	420	22	moderate sun + lights on	100%
4	100	415	28	strong sun + lights on	100%
5	100	412	26	Strong sun + lights on	100%

To show the performance of the filter, the measurements of the StarGazer before and after filtration are plotted as it could be seen in Fig 4.5. In the figure, the blue dots in (a, b, c) represent the raw measurements of the StarGazer sensor for X, Y and  $\theta$ . It can be seen that some measurements are totally far from the expected motion curve of the StarGazer.

Fig 4.5 (d, e, f) shows the results of applying the filter over these measurements. The yellow dots represent the raw measurements from the StarGazer sensor, while the blue dots represents the filtered measurements of the sensor. It can be seen clearly that there are some irregular points which are not located on the expected curve. The goal of the filter is to detect these wrong measurements and provide estimation for the StarGazer location.

It is clearly that the filter didn't follow the false measurements, but it detected them, and provided approximations for the position basing on the estimated measurements generated by the estimation equation, after neglecting the filtered measurements resulted from the update equation under these false values. Furthermore, it could be seen that when the sensor provides correct measurement, the filter use this measurement to update its state.

It is worthy to mention that the goal of the filter is not to improve the accuracy of the StarGazer sensor, but to detect the false measurements and provide an accurate estimation for the robot's position even when the sensor provides false measurements.

In [47], the implemented EKF failed in detecting the false measurements, even when these measurements have high localization error ( $>3m$ ). This is because the false measurements don't follow white Gaussian noise. In contrary, it could be seen that

the implemented Kalman filter is able to detect the false measurements, and provide accurate estimation for the location of the robot under these false measurements.

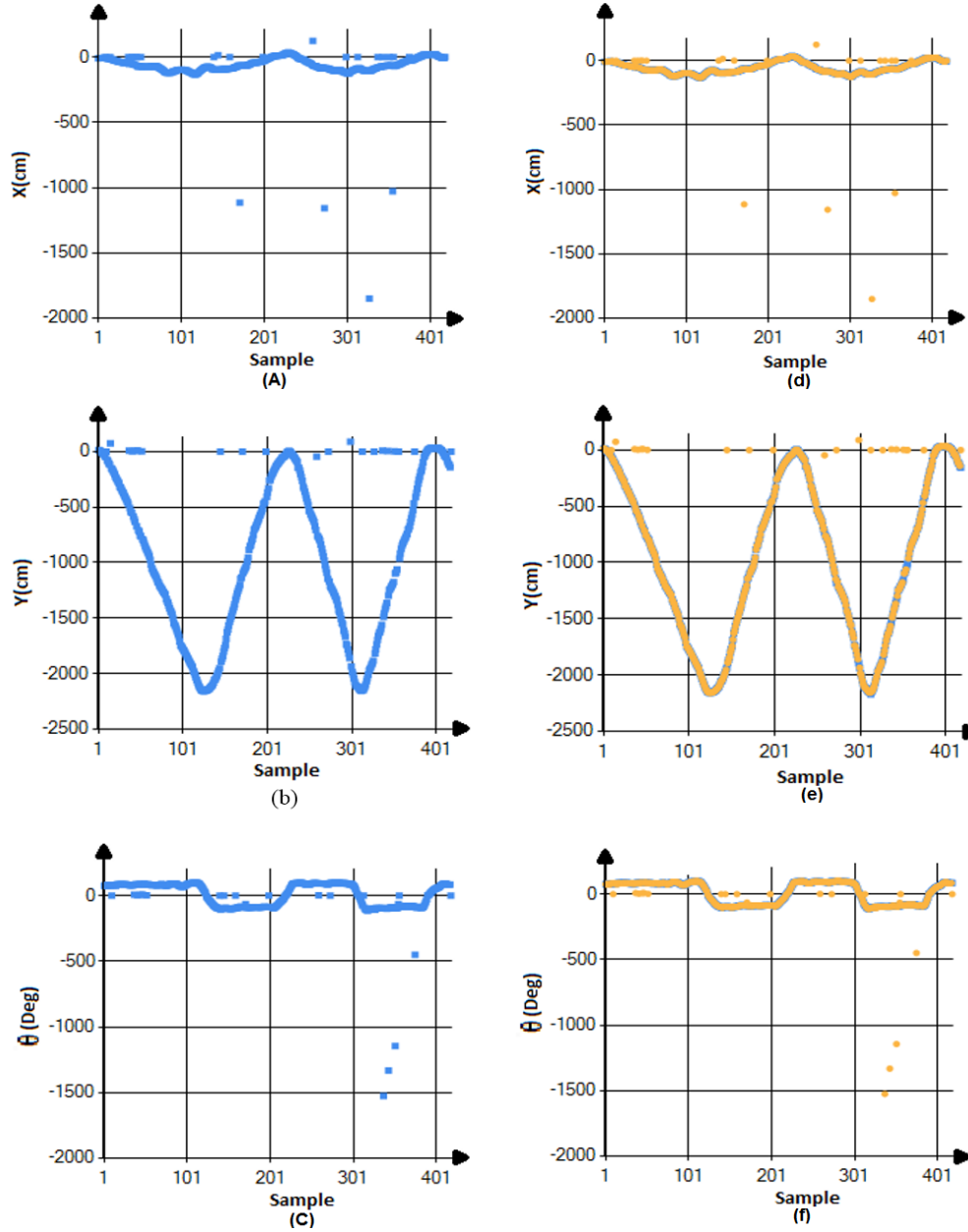


Figure 4.5: The StarGazer measurements before and after filtering: (a, b, c) The raw measurements of Stargazer sensor. The horizontal axis represents the number of raw measurements. The vertical axis represents the value of  $(X, Y, \theta)$  for (a, b, c) consequently. (d, e, f) The result of applying the proposed filter over the StarGazer sensor

## Chapter 5

# Human-Robot Interaction System for Indoor Mobile Robots

---

### 5.1 Introduction

In the future work environments, robot and human will work side by side in the same place, and this raises big challenges related to the ability of each of them from identifying the other, and interacting and exchanging the information when needed. Thus, robots must be equipped with a robust human-robot interaction system to be able from a secure work with the human.

Human-Robot Interaction is defined as the ability of a robot from distinguish a human, understand his orders or emotions, and implement a suitable response as a result of this interaction.

In recent years, the fast improvements in the field of artificial intelligence, machine learning and computer engineering, led to several forms of the interaction with human. It could be seen from the chapter 2, that the interaction formulation is based on the goal of the interaction, the tasks that the robot is doing, and the used sensors for the interaction.

In this work, the interaction will serve the task of human safety, by implementing a human-robot interaction system which enables the robot from recognizing the human and getting the motion information from him so both of them avoid the collision when they meet in narrow locations.

Kinect V2.0 sensor (Microsoft, USA) is widely used in applications, which require human and gesture detection [128]. This sensor has a RGB camera and an infrared camera. The combination of these two cameras provides the robot with a 3D vision of the work environment. The sensor uses time-of-flight to measure the distance between the sensor and the objects [129]. In this technology, the sensor sends infrared beam, and measures the time required by beam to travel back to the IR camera. The Kinect 2.0 provides a skeleton frame, which includes the skeletons for up to 6 humans to the robot. 3D dimensions for 25 joints express each skeleton. The sensor is able to detect the human within the distance between 0.8 to 4.5 meters.



## 5.2 System Description

It could be concluded in the literature survey, that the HRI systems are either designed to extract the emotions of human, receive the tasks from them via voice, gesture, or neurons, or to interact with human and learn from them how to execute certain tasks. Furthermore, it could be seen from the literatures that are related to collision avoidance, that the traditional collision avoidance systems are working autonomously, without getting any kind of feedback from the humans about their motion direction, and without a possibility to interrupt the motion of the robot when needed. Out of this a new robotic system which is based on HRI is implemented to serve the task of collision avoidance for mobile robotics [112].

In the implemented system, the robot will be equipped with a Kinect sensor to monitor the human, which exist in the navigation area, and get the gesture information from them. The person in front of the robot then will use gestures to inform the robot about the procedure, which it has to execute (stop, move forward, move backwards, go right, go left, resume). The robot will execute the orders based on the detected gesture and move to its final destination using global path planner system after avoiding the human [130]. To implement the human-robot interaction, seven gestures are used in this system. Table 5.1 depicts these gestures and the robot's response for each of them.

The Kinect sensor provides the robot with information about the human's joints. This information requires further processing to extract the corresponding orders issued by the human. Thus, it is required to implement a tool which is able to interpret the skeletal data into useful commands that the robot can understand. Since the interaction is based on human's arms, the data of five joints will be used to detect the gestures:

- The (y) values of the right and left elbows, and the right and left wrists. These values will provide sufficient information about the locations of the left and right arms in the space.
- The (y, z) values for the neck joint. These two values are necessary to train the model to classify the gestures of human with different heights.

Fig 5.1 shows the human joints extracted by the Kinect.

Table 5.1 The gestures used in the HRI system and their corresponding function

Gesture	Function
Stop	The robot will stop as long as the stop gesture is raised.
Move right	The robot will generate the collision avoidance path by selecting the closest region located to the right side of the master person.
Move left	The robot will generate the collision avoidance path by selecting the closest region located to the left side of the master person.
Move forwards	The robot will move toward the person as long as the “move forward” gesture is raised.
Move backwards	The robot will move backwards as long as the master person raises the “move backwards” gesture.
Master select	If several humans are in the path of the robot, the robot will not know to which person it has to interact, since two or more human might give different orders to the robot. To overcome this problem, “Master Select” gesture is assigned. When a person in a group raises the right arm 180°, the robot will know that it has to interact with this person and will ignore the gestures of other human in the work area.
Resume	The resume gesture is used when the master person orders the robot to ignore the human existence and complete its path. This is the case when the person thinks that the robot moves in a different path which he follows, or when the person will leave to another position far from the robot’s path.

To understand the gestures, a tool should be used which is able to map the joints’ coordinates to its classes. From literature survey, many methods are adopted to extract the gestures of human from Kinect sensor.

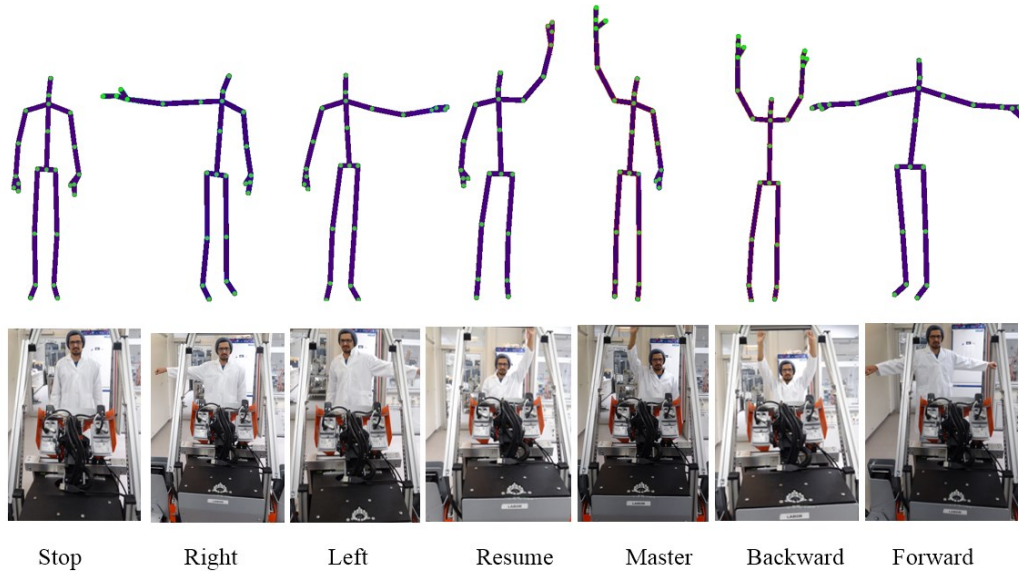


Figure 5.1: The gestures used for the interaction with the robot

It is also shown that the methods which are based on geometry analysis of the joints won't work correctly when the human plan is not parallel to the sensor plan, since the angles and the relative positions of joints will be different when the robot and human are not located face to face.

In real world, it is highly possible that the human is not adjacent to the robot, and the robot should still be able to interact with the human and understand their gestures correctly. Thus, it is expected that the methods which work on analyzing the data and classifying it basing on a previous training for these models could provide a better performance in identifying the corresponding order for a given gesture. In this study, two models are adopted to classify the skeletal information provided by the Kinect sensor. The first model is Support Vector Machine (SVM) [131] and the second model is Back Propagation Neural Network (BPNN) [132] [133]. These two models are trained and the performance is compared.

## 5.3 Support Vector Machine

### 5.3.1 Model Description

In SVM, the model works on searching for the line, which is able to separate the data optimally, in the current application, the data represents the measurements obtained from Kinect sensor. Supposing the data of the sensor are plotted in Fig 5.2, it could be seen that there is an infinite number of separation lines. Searching for the line, which is able to keep the largest margin between the two classes, does

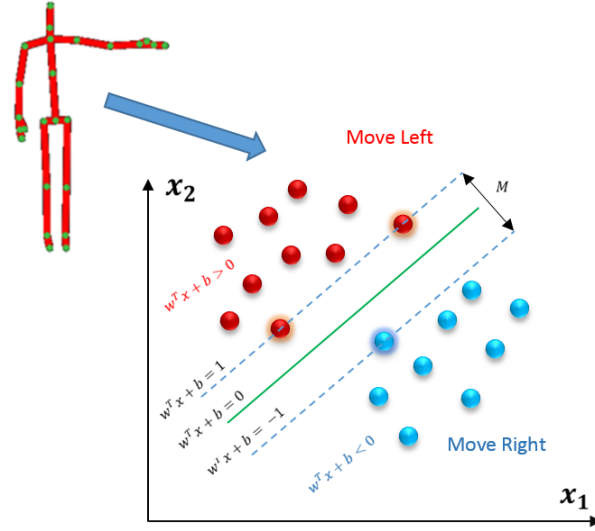


Figure 5.2: The Principle of Support Vector Machines

optimization [117] [134]. To find the mathematical representation of this problem, it is supposed that the separation line has the following equation:

$$f(x) = w^t x + b \quad 5.1$$

Where  $w^t$  represents the weight vector,  $b$  the bias and  $x$  input data.

Supposing that the line is able to separate the data into two classes:

$$wx_i + b \geq +1 \quad \text{if} \quad y_i = +1$$

For all points which are located at the first class and:

$$wx_i + b \leq -1 \quad \text{if} \quad y_i = -1$$

for the points which are located at the second class.

The margin  $\mathcal{M}$ , is defined as the distance between the positive class and the negative class.

$$(x^+ - x^-) = \frac{2}{|w^t|} \quad 5.2$$

The margin could be given as:

$$\mathcal{M} = \frac{2}{|w^t|} \quad 5.3$$

Optimization is done by searching for the line, which is able to keep the maximum margin between classes without any error in classifying the training data. This is equivalent to maximizing the margin, or minimizing the  $\varphi$  since:

$$\varphi(w) = \frac{1}{2} w^t w \quad 5.4$$

Subject to

$$y_i(wx_i + b) \geq 1 \quad \forall i \quad 5.5$$

Support vectors are defined as the training vectors, which are located on the auxiliary hyperplanes as it could be seen in Fig 5.2 and could be represented mathematically as:

$$f(x) = w^t x^+ + b = +1$$

$$f(x) = w^t x^- + b = -1$$

Usually, finding an optimal hyperplane which is able to separate the whole data is difficult, since some vectors could be correctly classified but not located within the boundary hyperplanes, or it could even be misclassified. Thus a slack variable ( $\xi$ ) is defined which represents the distance between the misclassified vector and the correct boundary hyperplane [117]. For the classification models which includes misclassified vectors we define:

$$y_i(wx_i + b) \geq 1 - \xi_i \quad : i = 1, 2, \dots, n \quad 5.6$$

$$\xi_i \geq 0 \quad \forall i = 1, 2, \dots, n$$

Thus, for the vectors which are correctly classified and within the boundary hyperplane  $\xi = 0$ , and for the vectors which are correctly classified but they are not located within the boundary hyperplane  $0 \leq \xi \leq 1$ , while  $\xi \geq 1$  for the vectors which are misclassified. Thus the optimization problem takes the form:

$$\min ||w||^2 + C \sum_i \xi_i \quad : i = 1, 2, \dots, n \quad 5.7$$

Where  $C$  is the penalty factor, which controls the trade-off between maximizing the margin and minimizing the training error. Large  $C$  means that the model is focusing on minimizing the error rather than maximizing the margin, while small  $C$  means the model is focusing on maximizing the margin with less care about minimizing the training error.

The penalty factor  $C$  is selected manually, so to test the performance of the model for different penalty factors, training algorithms are needed, which are able to train

and test the model. Many training algorithms existed for this purpose as exhaustive cross validation, Repeated Random sub-sampling validation, and K-fold crosses validation. In this research, K-fold cross-validation algorithm is used for training the model and selecting the appropriate penalty factor [135], [136].

### 5.3.2 Model Training

To put the SVM into real applications, the model has to be trained given a set of training inputs and the desired outputs which the model has to generate as a response to the applied input. Thus, to train the model, a training set is required to tune the parameters of the model. It could be seen from Fig 5.1 that the measurements of the five joints in the body frames are extracted. For the right and left wrists, and the right and left elbows, the (y) values are extracted. Furthermore, the (y, z) values of the neck are also extracted.

To build the training set, the gestures of four people with different heights are taken; each person is asked to implement the whole 7 gestures in 16 different positions in distances with range [1.8, 4] m. Thus, for each person there is  $16 \times 7 = 112$  training samples, and the total training set is equal to 448.

Since it is possible that the human is not totally facing the robot, the training samples are taken with deviation angles  $[-40^\circ, 40^\circ]$  between the participants and the Kinect level. By training the model with training samples that are taken with different angles of people, it is expected that the HRI will be able to recognize the gesture of human even if they are not located face-to-face with the front plane of the Kinect sensor.

To train the SVM, k-fold cross validation algorithm is used. K-fold cross validation is a training algorithm which is used for training and evaluating the trained classifier given the available set of training data. In this algorithm, the data set  $D$  is divided into equal subsets  $K_i$ . Each subset  $K_i$  includes  $\frac{D}{K} = m$  samples of the training set:

$$D = \sum_{i=1}^n K_i : i = 1, 2, \dots, k \quad 5.8$$

Since each  $K_i$  represents a subset of the dataset  $D_n$ . Fig 5.3 shows the flow chart of training the SVM model.

In the next step, the data subsets  $K_{1,2,\dots,i-1}$  are used for training the model  $M$  with the given set of parameters; while the remaining set  $K_i$  is used to evaluate and validate the model, and the number of misclassified samples for the set  $K_i$  is computed.

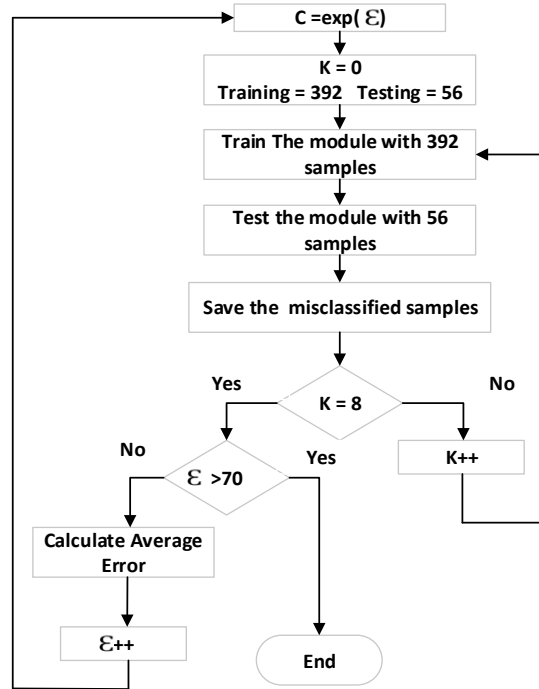


Figure 5.3: The flow chart for building and training the L-SVM model

In this work,  $k = 8$ . Fig 5.4 shows the subsets distribution for training both the SVM and BPNN models.

K=0	Test=56	Train = 392	
K=1		Test=56	Train = 392
K=2		Test=56	Train = 392
K=3		Test=56	Train = 392
K=4	Train = 392		Test=56
K=5	Train = 392		Test=56
K=6	Train = 392		Test=56
K=7	Train = 392		Test=56

Figure 5.4: The k-fold sets for the L-SVM model

## 5.4 Back Propagation Neural Network

### 5.4.1 Model Description

Back-Propagation neural network is a multi-layer network that is composed of input layer, hidden layers and output layer. Each layer is composed of several neurons;

each of these neurons in a certain layer is linked to each neuron in the following layer via weighted links. Furthermore, each neuron in the hidden and output layers has an activation function, which calculates the output of the neuron after receiving the weighted inputs from the neurons in the previous layer. Fig 5.5 shows the hierarchy of the BPNN.

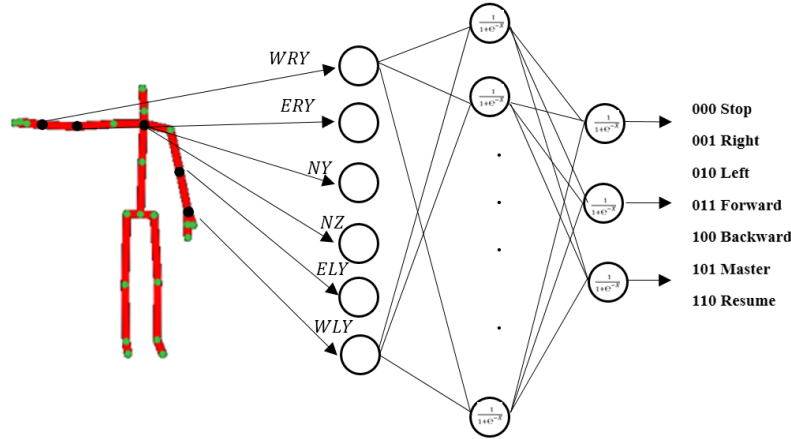


Figure 5.5: The hierarchy of the BPNN

In the training step, the weights are selected randomly with small values,  $w_{i,j} \in [-0.5, 0.5]$ . Then the training vector is applied to the network's inputs, and propagate these inputs via the layers to the output layer. The error of the network is calculated as the deviation of the network output from the desired output. This error is propagated inversely to update the weights of the layers.

Considering a three layer BPNN,  $w_{ij}$  represents the weight of the connection between the neuron  $i$  in the input layer, and the neuron  $j$  in the hidden layer;  $w_{jk}$  represents the weight of the connection between the neuron  $j$  in the hidden layer, and the neuron  $k$  in the output layer [132]. The input and the output of a neuron in the hidden layer is given as:

$$x_j = \sum_{i=1}^n x_i \cdot w_{ij} - b_j \quad 5.9$$

$$y_j = \frac{1}{1 + e^{-x_j}} \quad 5.10$$

Where:

$b_j$ : the bias for the neuron



$n$ : the number of inputs for the network

Actually, the right term of the output equation represents a sigmoid function which is used widely in the BPNN for activating the neurons of the hidden and output layers. For calculating the output of the network, the same steps are followed using the equations 5.9 and 5.10.

After completing the feed forward step, the model error is calculated for each output neuron. The output error represents the difference between the desired output and the actual output:

$$e_k = d_k - y_k \quad 5.11$$

After calculating the error, the weights have to be corrected, to decrease this error. Thus, the updated weights between the hidden and output layers is given as:

$$w(new)_{jk} = w(old)_{jk} + \Delta w_{jk} \quad 5.12$$

Where  $\Delta w_{jk}$  is the weight correction, and it is calculated as

$$\Delta w_{jk} = \eta \times y_j \times \delta_k \quad 5.13$$

Where  $\delta_k$  is the error gradient which represents the derivative of the multiplication of activation function with the error of the output neuron [133]:

$$\delta_k = \frac{\partial y_k}{\partial x_k} \times e_k \quad 5.14$$

Considering using the sigmoid activation function:

$$\delta_k = y_k \times (1 - y_k) \times e_k \quad 5.15$$

The updated weights will be given as:

$$w_{jk(new)} = w_{jk(old)} + \eta \cdot y_j \cdot \delta_k \quad 5.16$$

In similar way, the error gradient of the neurons in the hidden layer is given as:

$$\delta_j = y_j(1 - y_j) \sum_{k=1}^n w_{jk} \cdot \delta_k \quad 5.17$$

Then, the weight corrections are given as:

$$\Delta w_{ij} = \eta \times x_i \times \delta_j \quad 5.18$$

And the new weights of the hidden neurons will be calculated as:

$$w_{ij(new)} = w_{ij(old)} + \Delta w_{ij} \quad 5.19$$

Where  $\eta$  is the learning rate, and it is taken as a positive value less than 1.

### **5.4.2 Model Training**

In BPNN, the goal of training the model is to find the best number of hidden neurons, with the best weights values that provide the minimum output error. The model will have six inputs; one for each joint's value, and three outputs which provide the gesture to the robot. To find the best number of hidden neurons, several models have been trained and tested with number of hidden neurons  $n = [3, 21]$ .

Similarly to SVM model, the BPNN is trained using the same training data set (448 samples). Furthermore, the k-fold cross validation with  $k = 8$  will be used to train the model and test it for different number of hidden neurons.

## **5.5 System Implementation**

### **5.5.1 Kinect Position**

Kinect V2 sensor has  $70^\circ$  horizontal and  $60^\circ$  vertical fields of view. Thus, the positioning of the sensor has an important role in detecting the human in different positions. The next chapter will show that the robot is required to interact with the human and execute the procedures of collision avoidance when the distance between the robot and a person is around 2m. Thus, the Kinect has to be fixed in a position that allows the robot to detect the human within the distance 2-4m. Furthermore, it should be taken into consideration that the robot has to detect the static obstacles on the floor of the navigation area, which could be a future work for the implemented system. Thus, it is decided to fix the Kinect sensor on a high 75cm above the floor. Fig 5.6 depicts the position of the Kinect and the vertical distances that the robot can detect within the given position. It could be seen that for the distance 2m, the robot can detect the human with heights around 2m.

### **5.5.2 Human-Robot Interaction System Description**

In the previous sections, it is mentioned that the robot interacts with the human using seven gestures. Since it is possible to have several human in the path of the robot, it is probable that each person gives a different order to the robot. Thus, the robot won't be able to understand what it has to do. To overcome this problem, a special gesture "Master Select" is allocated to enable the robot to detect the master

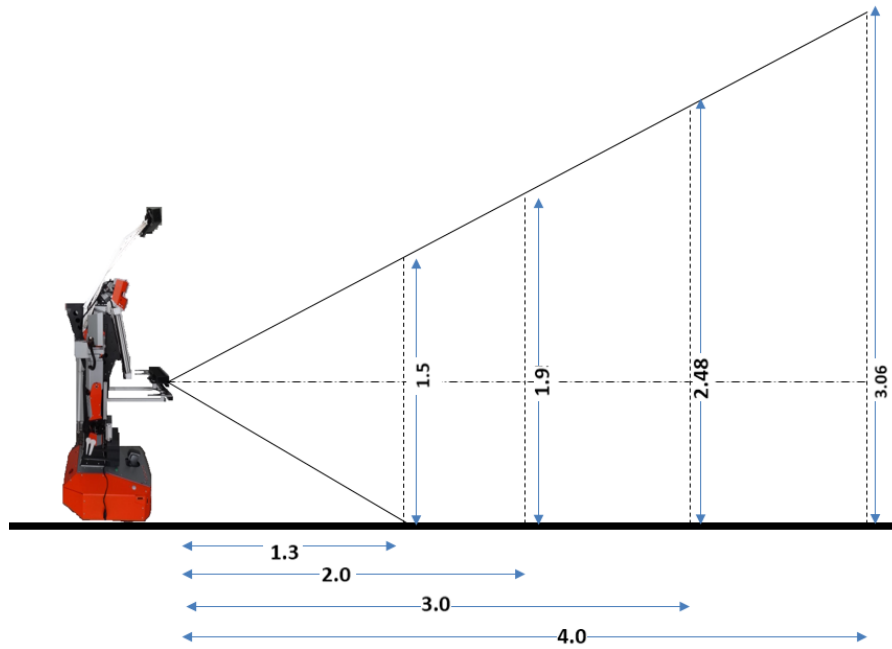


Figure 5.6 Position of Kinect sensor and detection range

person. When the master person is detected, the robot will limit the interaction to this person and ignore the gestures of other human in the area.

The Kinect sensor provides 30fps, this is a high rate for the interaction, since it is not possible to the robot and the human to implement the interaction with this rate. Moreover, if the interaction is done in a short period of time, it is possible that the robot mistakenly interacts to the arms' position while the master person is still moving them. For example, the robot might move forward "two arms horizontally" while the master person is still moving his arms to reach the move backward order "two arms vertical". To overcome this problem, the gestures of each person were stored in a unique register. Each register records 6 gestures for each person. If the whole values of the register are the same, then the robot will execute the saved gesture in the register. This will guarantee a stable interaction with the arms' movements. When a person moves away from the robot's path, the assigned register will be cleared automatically. Since the robot compares each 6 gestures, the human has to fix his arms' position for a given gesture for at least 200 ms. Fig 5.7 shows the flow chart for the human-robot interaction system.

Once the robot detects human in the path, it will save and update the positions of human and their distances from the robot, this data is important for the collision avoidance system to be able to calculate the required path soon when an order is assigned. The robot will then search for a master, by classifying the gestures of all human in front of it, and compare the contents of each register to check whether a

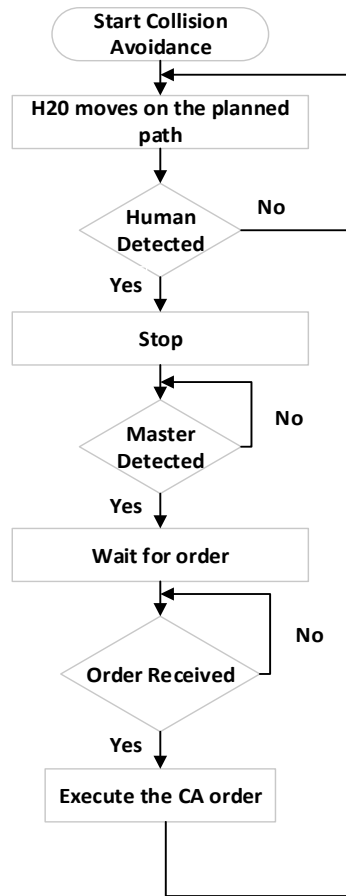


Figure 5.7: The control of the robot using Human-Robot Interaction

master is existed. When a master is detected, the robot will keep updating the distances of each person to keep the CA system has the positions of each person, but it will limit the interaction with the master person. The system will then classify the gestures of the master person, and forward them to the collision avoidance system for executing the requests.

### 5.5.3 Sensor False Inferred Data

In Kinect sensor, when the body is located near the vision limits of the sensor, it will provide an inferred estimation of the joints' positions of the body. Actually, the experiments show that this estimation is not accurate, and this leads to a false detection for the gestures when the Kinect provides false inferred joints to the HRI system. Fig. 5.8 shows an example of a false inferred skeleton of the body. The inferred joints are those which are marked with yellow dots. It is clear, that the gesture will be misclassified when the Kinect provides these false joints'

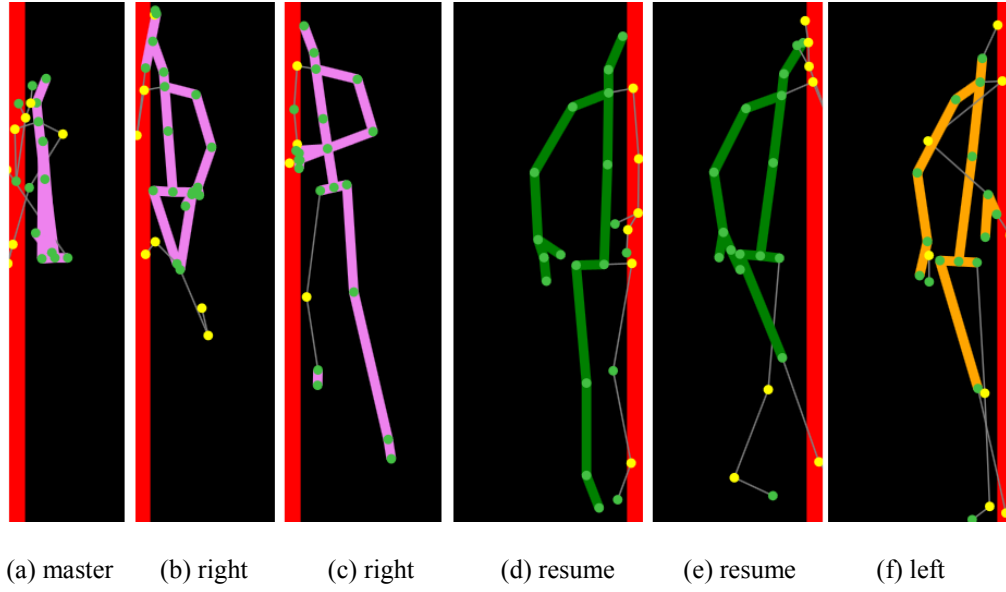


Figure 5.8: The false skeletal joints due to angle view limits of Kinect

measurements to the classifier. To overcome this problem, the system will check the status of each joints of interest (neck, elbows, wrists), and when there are more than two inferred joints, the measurement won't be forwarded to the classifier as long as the joints are not correctly detected.

## 5.6 Experimental Results

The experiments were done on two levels; the first experiments show the training results of the BPNN and SVM, including the time required from each model to implement the training, and evaluating the performance of each model in classifying the samples. In the second experiment, the HRI model is tested in real environment, and five humans with different physical shapes tested the human-robot interaction system with different positions and angles.

### 5.6.1 Training the SVM

In this work, a linear-kernel support vector machine model is used. To obtain the best model of the SVM, the model is tested over a wide range of penalty values:

$$C = e^{\varepsilon} \text{ where } \varepsilon = -10, -9, \dots, 69, 70.$$

Fig 5.3 showed the flow chart for training the model. Model training is done using EMGU library, over a Laptop with the processor Intel i7-3537U (2.0 GHz with 4 CPUs) and 8 GB RAM. Appendix 3 shows the complete experiments over the model, while table 5.2 summarizes the training results for the model.

Table 5.2: The training results for the L-SVM model

C	Total Error	Average Error	Success Rate (%)	Train Time (ms)	Test Time (ms)	No. Support Vectors
$[4.5 \times 10^{-5}, 0.0067]$	379	47.3	15.53	[103,112]	<1	21
0.0183	355	44.3	20.9	102	<1	21
0.0497	156	19.5	65.17	88	<1	21
0.135	33	4.12	92.64	71	<1	21
0.367	8	1	98.2	68	<1	21
1, 2.718	9	1.125	97.99	49, 56	<1	21
<b>7.389</b>	<b>8</b>	<b>1</b>	<b>98.2</b>	<b>45</b>	<b>&lt;1</b>	<b>21</b>
20.08, 9.2 $\times 10^{29}$	16	2	96.4	[41, 53]	<1	21

In table 5.2, the total error represents the total misclassified samples for the complete folds, while the average error represents the total error divided by the number of folds (here 8). To estimate the performance of the SVM model for the given penalty value, the success rate is evaluated as:

$$Success\ Rate = 100 - \frac{average\ error}{number\ of\ test\ samples} \times 100 \quad 5.20$$

Furthermore, the train time represents the total time required to train the model for the whole 8-folds for the given penalty value. The test time, represents the time required from the trained SVM model to test a given sample. It could be seen that the test time for any sample after training the model is less than 1ms.

From the table 5.2, it could be seen that setting the penalty to small values less than 0.13, will lead the model to show poor classification for the training sets. On the other hand, the values of the penalty in range [0.367, 7.389] show the best performance with eight to nine misclassified samples. Moreover, the table shows that using higher penalty values won't improve the performance of the SVM model, since the model fails in classifying 16 test vectors. It could also be concluded that after training the model, the total time required to predict a given sample is less than 1ms. From the appendix 3, it could be seen that the average training time for the whole experiments is equal to 50.95ms.

By comparing the experiments in the table 5.2, it could be seen that the best performance of the SVM model is with penalty value  $c=7.389$ , with only eight misclassified samples, which is equal to a success rate 98.2%.

### 5.6.2 Training the BPNN Model

Fig 5.4 showed the general structure of the BPNN model. The model has six inputs which represent the y-values of right and left wrists and elbows, besides to the (y, z) values for the neck. Since the model has to distinguish seven gestures, three outputs were selected for the model, as it could be seen from the table 5.3.

Table 5.3: The outputs of the BPNN for the given gesture

	<b>Q1</b>	<b>Q2</b>	<b>Q3</b>
Stop	0	0	0
Move Right	1	0	0
Move Left	0	1	0
Move Forward	1	1	0
Move Backward	0	0	1
Master Select	1	0	1
Resume	0	1	1

To search for the best number of hidden neurons, the model is tested using the same training set and the same k-fold algorithm which were used to train and optimize the SVM model. The optimization is done by searching for the best number of hidden neurons for the model, and the best weights values. Thus, the neurons are selected in the range 3 to 21. Table 5.4 summarizes the training results.

From the table 5.4, it could be seen that the BPNN could provide very satisfied results for any number of hidden neurons. The model is able to classify the testing sets with a successful rate equal to 100% with a classifying time less than 1ms. While the average training time is equal to 117.2 ms.

Table 5.4: The result of training and testing the BPNN for several hidden neurons

No. of Hidden Layers	No. Errors	Train Time (ms)	Test Time
3	0	101	<1
4	0	115	<1
5	0	112	<1
6	0	114	<1
7	0	115	<1
8	0	103	<1
9	0	107	<1
10	0	109	<1
11	0	112	<1
12	0	123	<1
13	0	112	<1
14	0	129	<1
15	0	131	<1
16	0	128	<1
17	0	124	<1
18	0	111	<1
19	0	136	<1
20	0	126	<1
21	0	120	<1

### 5.6.3 Comparison between SVM and BPNN

In comparison between the tables 5.2 and 5.4, the following results could be concluded:

- The BPNN could classify the whole test sets successfully despite of the number of hidden neurons.
- The L\_SVM model showed better performance for penalty values close to 1. Still, whatever the penalty value, it couldn't classify the whole patterns successfully, with the best performance of the model is at  $c = 7.389$ , with 8 misclassified samples.
- The L\_SVM showed faster training than the BPNN with average training time 50.95ms for the L\_SVM and 117.2 ms for the BPNN.
- Both models were able to classify an input vector with less than 1ms.

Thus, the BPNN with 8 hidden neurons is chosen for gesture recognition in the proposed Human-Robot Interaction System.



### 5.6.4 Human-Robot Interaction System Test

After implementing the HRI system with the BPNN, it is required to test the system over real work conditions. Fig 5.9 shows the test area for the robot and the positions which the humans whom implemented the tests were located. Five humans with different physical shapes (heights, widths, clothes...) were asked to implement the tests. Each person is asked to implement a certain gesture in each position, with a deviation angle  $(-40, -20, 0, 20, 40)$ . The complete experiments could be found in appendix 4, while the table 5.5 summarizes these experiments.

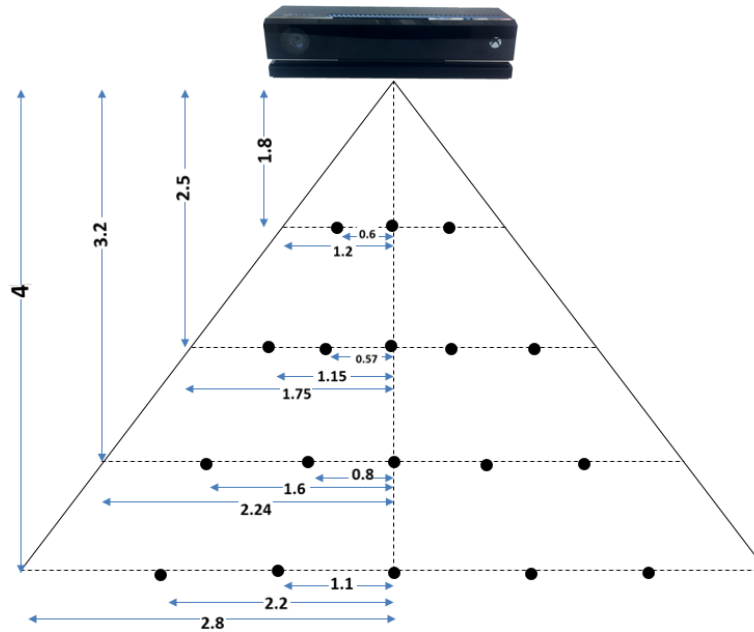


Figure 5.9: Representation of the test environment (dimensions in meter)

By analyzing the experiments in the table, the following results could be outlined:

- The BPNN classifier shows 100% correct classification for the whole gestures provided to it.
- Out of 90 experiments, the human-robot interaction system misclassified two gestures. These two gestures are misclassified because of a false positions of the joints which are provided by the Kinect sensor to the classifier.
- Out of 90 experiments, it shows that 84 experiments needed a processing time in the range  $[337, 396]$  ms, while there are six experiments required processing time in range  $[735, 773]$  ms. The reason that the six experiments required almost double time that a user in the experiment didn't fix his arms during the experiment, and this lead from the HRI system to clear the register assigned to the user and fill it again with new set of classified measurements.

Table 5.5: The summary of experiments over the HRI system

Person	Number of experiments	Height	False gesture	State transitions
1	18	165	1	0
2	18	170	1	3
3	18	177	0	1
4	18	179	0	1
5	18	188	0	1

By comparing the current results with the other methods which were discussed in the literature survey, it could be seen that the proposed human-robot interaction system is easier and more applicable compared to other methods using touchable and wearable sensors since it is not required to attach sensors to human to be able to interact with the robot. Moreover, the system could be used in social work environments easily since there is no need for further devices to be fixed on each person existed in the same navigation area of the robot.

Moreover, by comparing the performance of the implemented HRI system, with the other HRI systems which were based on gesture recognition (table 2.6), it could be seen that the previous systems were implemented on the concept that the robot and the human are located face-to-face, while in the realized system, the robot can still recognize the gestures successfully even when a human is deviated with  $\mp 40^\circ$  from the straight sight between the robot and the human. While the other methods which were based on geometry will fail when the person is not located face to face with the Kinect plan.

In table 2.6 it is shown that the success rate of several methods ranges between 83.33% to 98.4%. While the success rate of the implemented SVM is 98.2% and 100% for BPNN. Despite of that the Hidden Markov Model shows a bit higher success rate (98.4%) than the SVM, the implemented SVM model can recognize people even when they are not directly facing the robot. Thus, it could be concluded that both classifiers SVM and BPNN show higher classification performance than other methods which are used in literature.

Finally, it could be seen that the only limitation from the human-robot interaction system comes mainly from the sensor itself. The experiments which were implemented over the system showed that there were two misclassified gestures. This is because the sensor provided wrong joints' dimensions to the BPNN model. Moreover, it is noticed that the Kinect sometimes doesn't detect the body, and it requires from the user to move a bit, or shake his body to allow the Kinect from

detecting him. The user can know that the Kinect didn't detect him by monitoring the robot's screen which shows the skeletal frames of each detected person.

## Chapter 6

# Collision Avoidance System for Indoor Mobile Robots Basing on Human-Robot Interaction

---

### 6.1 Introduction

In future work environments, robots will work alongside to humans, and this raises challenges related to the robustness of these robots in detecting human, interacting with them and avoiding physical accidents to them. Thus, any robotic system must be equipped with a robust collision avoidance system which enables the robot to detect the obstacles and avoid them, especially the human.

In chapter 3, it could be seen that many collision avoidance systems have been implemented for mobile robotics. The general concepts of these systems are summed up by detecting the obstacles' distribution in the work area, extracting the regions between these obstacles that are wide enough for the robot to pass, and selecting the region which is closest to the goal direction.

Despite of the good performance of many of these methods, it has some short comes especially when robots navigate in social environments which include many human moving and sharing the same working area of the robot as in laboratories, restaurants, and hospitals.

When the robot moves in social environments, it is possible that a group of human are located on the same path of the robot. Thus, the robot will try to avoid those humans by adjusting its path several time to avoid collision with them as it could be seen in fig 6.1. This will cause the robot to move in the middle of this group in curved paths, which will cause the human to get confused, besides to increase the time required by the robot to reach its goal.

Furthermore, it is possible to have a bottleneck problem when the human and the robot are located in narrow locations such as corridors, so neither of them is able to avoid the other.

Additionally, the performance of the collision avoidance system is merely related to the sensor's detection range. Thus, in some situations, it is possible that the human which is existed near to the robot to have more information about the obstacles' locations in the path of the robot, and if this human is able to inform the robot about the best path it has to follow, this will improve the performance of the collision avoidance and avoid the situations as in Fig 6.1.

Out of this, a new collision avoidance system is proposed which takes into consideration the human as an intelligent-moving obstacle [113]. The proposed system will allow the human to interact with the robot and provide it with the suitable direction of the collision-free path when the human finds it is necessary.

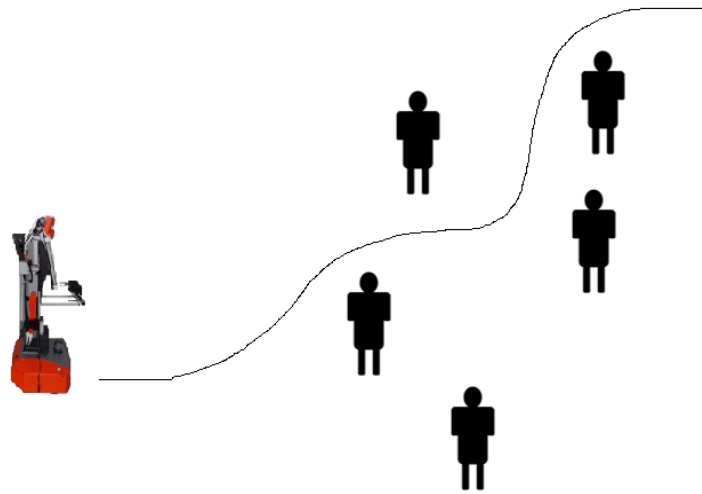


Figure 6.1: Robot motion between a group of humans

In this system, when the robot meets humans, it will ask them to interact with it via voice requests. If the user is interested in mastering the motion of the robot, he can manage it via interaction using the human-robot interaction system described in the previous chapter.

Thus, the user can order the robot to move backward/forward in case of narrow corridors, or he can select the direction of the path that the robot has to move, so the user moves to the other direction. On the other hand, if no human interacted with the robot for a certain period of time, the robot will calculate its local path autonomously taking into consideration selecting the path that is closest to its original path. The method is called cooperative collision avoidance (CCA) since both the robot and the human share the responsibility of avoiding each other [112] [113].

## 6.2 System Description

The proposed system gives mutual responsibilities for the robot and humans to avoid each other and search for the safe paths via interaction. The robot will interact with the human by sending voice messages to them, and receiving the responses from the human via the human-robot interaction system which is described in the chapter 5.

When the robot detects human(s) in its path for distances  $d > 2\text{m}$ , it will warn them by sending a voice message “Robot is coming”, to notify the humans that they are in the path of the robot. In this case, the humans will be aware of the existence of the robot, and they can either move away from the path of the robot, or be ready for interaction.

If the human moved away from the path of the robot, it will continue its path to the goal location. Else, if the human stayed in the path, and the distance between the robot and the closest person is less than 2m, then the robot will stop and send voice message “Interact”. In this case, the robot gives the option to the human to either cooperate in avoiding each other via interaction, or to implement the collision avoidance autonomously. The robot gives a certain period of time to the human to interact “3 seconds”. Thus, if a person raised his right arm vertically “Master Select gesture” within the 3 seconds, the robot will keep stopped waiting for the next order from the master user. If the period is passed, the robot will then implement the collision avoidance path autonomously. After executing the collision-avoidance procedures, and if there is no more human in the path of the robot, it will complete its path to the goal location using the multi-floor navigation system [130]. Fig. 6.2 shows the flow chart for the proposed system.

## 6.3 Collision Avoidance System

In either the cooperative or autonomous collision avoidance methods, the robot has to find the free spaces that it can move between the humans without collision. Thus, a robust collision avoidance system is developed which takes into consideration the width of the humans and their distributions in the navigation area.

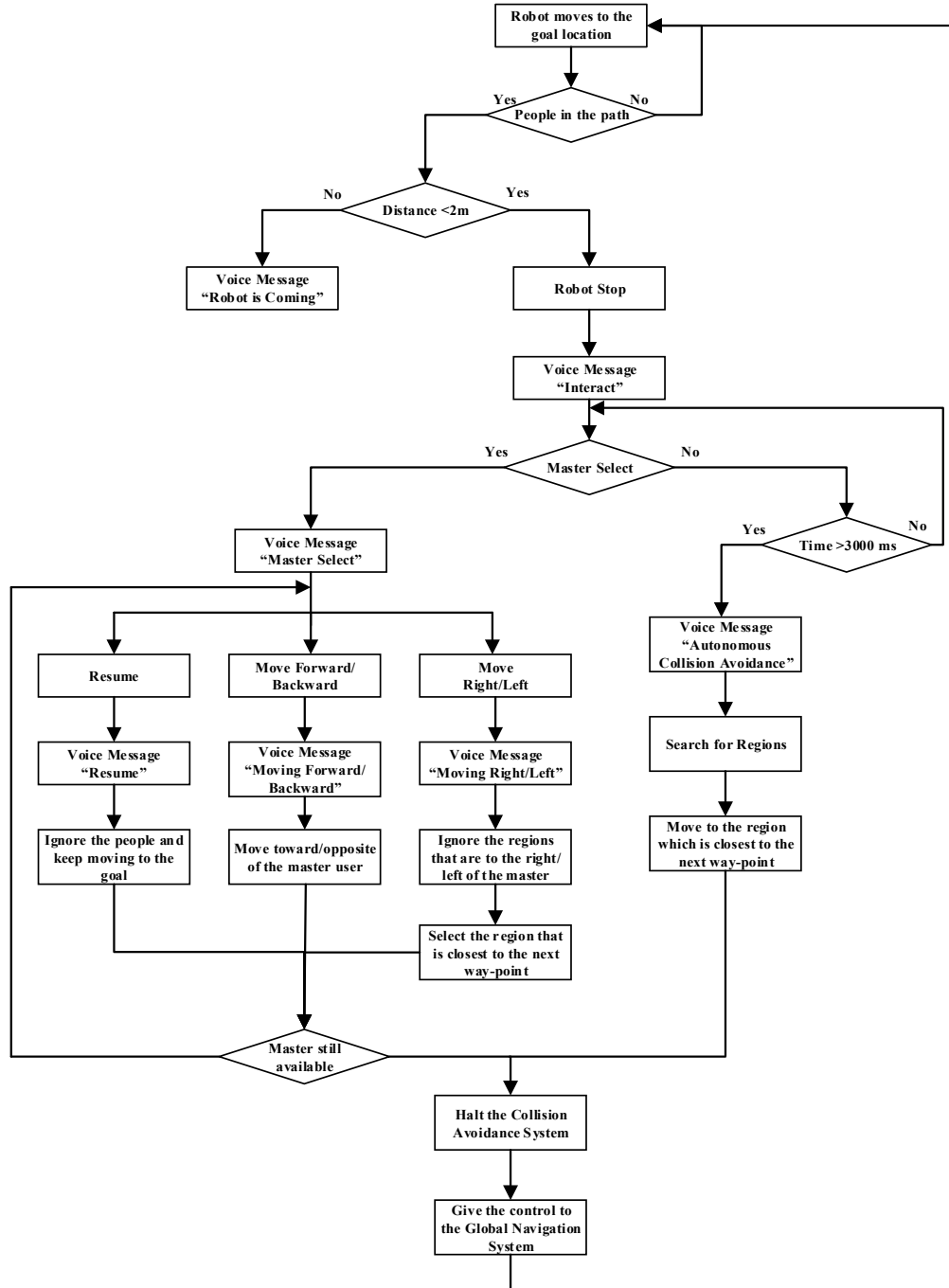


Figure 6.2: The flow chart of the collision avoidance system

In this system, the robot detects the positions of the human in its path, and specifies the distance of each person from it using the Kinect sensor. Furthermore, the width of each person is calculated by measuring the distance between the right and left shoulders of the human. Thus, the width of each obstacle “person” is given as:

$$r_p^i = |r_{rs}^i - r_{ls}^i| \quad (6.1)$$

Where:

$r_p^i$  the radius of the person  $i$  in the group

$r_{rs}^i, r_{ls}^i$  the  $x$  coordination of the left and right shoulders

The robot will search for the available regions which compose a potential space for implementing the collision-avoidance path. The robot will distinguish between the middle regions, which are the free spaces between the humans, and the terminal regions which are the regions to the left and right of the terminal human in the group.

For the middle regions, the robot will check the width of the regions between human:

$$R_{av}^i = (X_p^{i+1} - X_p^i) - (r_p^{i+1} + r_p^i) \text{ where: } i = 1, \dots, n-1 \quad (6.2)$$

Since  $n$  represents the number of humans detected by the Kinect sensor.

Furthermore, the robot estimates the minimum region width required for the robot to pass without collision:

$$R_{\min(mid)}^i = 2 \times r_r + r_p^{i+1} + r_p^i + d \quad (6.3)$$

Where:

$r_r$  robot's radius.

$d$  the safety distance around the robot.

Fig 6.3 shows the calculation of the middle regions.

In the next step, the robot checks the possibility of generating the avoidance path to the right and left of the most right/left human in the group. This calculation is done by considering the maximum detection angle of the sensor ( $70^\circ$ ), so the robot will compare the width of the region between the terminal person and the last point that the sensor can detect for the given depth, as it could be found in Fig 6.4, which shows the calculation of the region for the right-terminal person. To calculate the width of the right terminal region, the following equations are used:



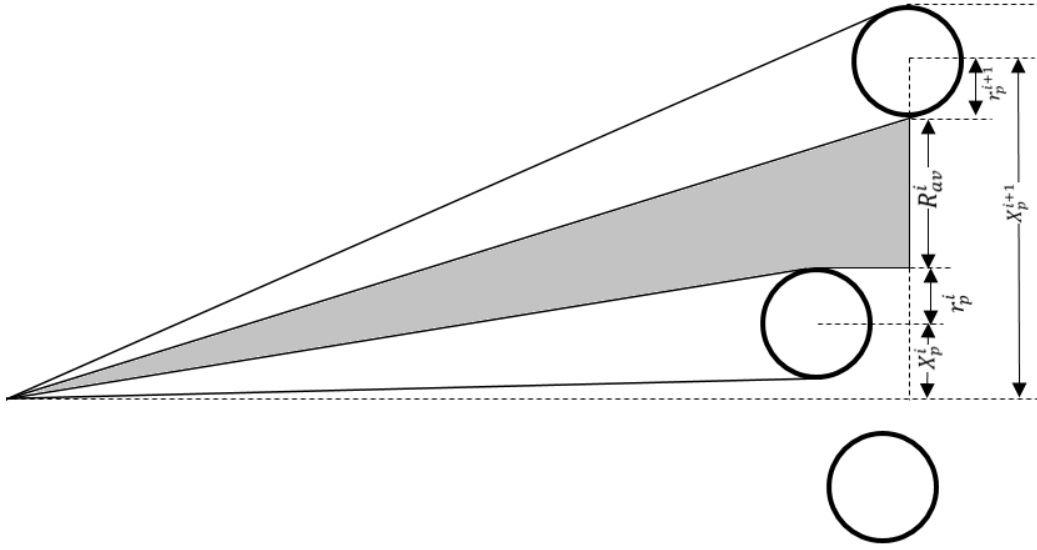


Figure 6.3: Calculation of middle region

$$X_{term}^n = Z_p^n \cdot \tan 35^\circ \quad (6.4)$$

$$R_{av}^n = X_{term}^n - (X_p^n + r_p^n) \quad (6.5)$$

Similarly, the following equations are used to calculate the left terminal region:

$$X_{term}^0 = Z_p^0 \cdot \tan(-35^\circ) \quad (6.6)$$

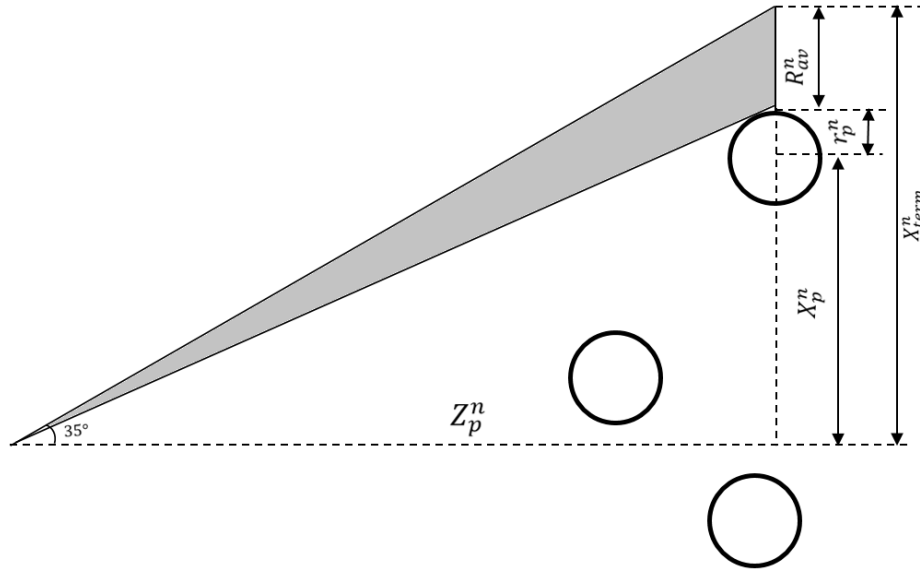


Figure 6.4: Calculation of the terminal region

$$R_{av}^0 = (X_p^0 - r_p^0) - X_{term}^0 \quad (6.7)$$

After receiving the available terminal regions, the robot checks whether these regions are wide enough to navigate or not using the following equations:

$$R_{min}^n = r_r + r_p^n + d/2 \quad (6.8)$$

$$R_{min}^0 = r_r + r_p^0 + d/2 \quad (6.9)$$

The candidate regions are the regions that are wider than the minimum required width which the robot needs to generate the collision-free path safely:

$$V = \sum_{i=1}^{n-1} R_{av}^i + R_{av}^0 + R_{av}^n \quad \text{if } R_{av}^i > R_{min}^i, R_{av}^0 > R_{min}^0, R_{av}^n > R_{min}^n \quad (6.10)$$

After detecting the whole candidate regions that the robot can go through, it will select the region based on the collision-avoidance method (cooperative or autonomous). In cooperative collision avoidance, the region selection will be based on the motion direction that the master person guided the robot to move through, while in autonomous collision avoidance, the region selection is based on the direction of the next way-point obtained from the global navigation system.

As it is mentioned before, when the robot detects humans, it will warn them to move away from its path. If the distance between the robot and the user is less than 2m, the robot will stop for three seconds to guarantee the safety of the human. If a user raised his/her right arm 180° within the given period, the robot will implement the movement based on the orders issued by the master user via interaction. Thus, it will move forward/backward or will execute the cooperative collision avoidance when the user activates the move right/left gestures. If no human interacted with the robot, it will consider that the human delegated it to select the collision-free path, so it will implement the autonomous collision avoidance. In the following, the two collision avoidance strategies are explained in details.

### 6.3.1. Cooperative Collision Avoidance based on Human-Robot Interaction

In cooperative collision avoidance, both the master user and the robot will cooperate in finding a secure collision-free path via interaction.

When the distance between the robot and a person is less than 2m, the robot will ask the human to interact using the voice request “interact”, and it will stop for three seconds allowing the human to interact with it. Within the stopping time, the user who wants to interact with the robot has to raise his right arm 180°, notifying the robot that it has to interact with him. The robot will then execute one of the following actions:

- Move forwards/backwards: The user can use the move forward/backward by raising his both arms 90°, and 180° respectively. In this case, the robot won't search for the regions or generate collision avoidance path. Instead it will obey the orders from the master user and move forwards / backwards as long as the master is still raising these gestures. This action is very necessary to avoid the bottleneck problem when the user and the robot meet each other in narrow areas, so neither of them is able to avoid each other due to the lack of free space as it could be seen in Fig 6.5. Moreover, this action is necessary when the user wants to guide the robot to move in dense and cluttered area. Thus, the user can order the robot to move either forward or backward to another free walking area, then he can either pass the robot, or can order it again to move to its left or right. The robot in these orders will move for 1.2m, and implement another reading for the gesture. If the gesture is still raised, it will implement additional 1.2m.

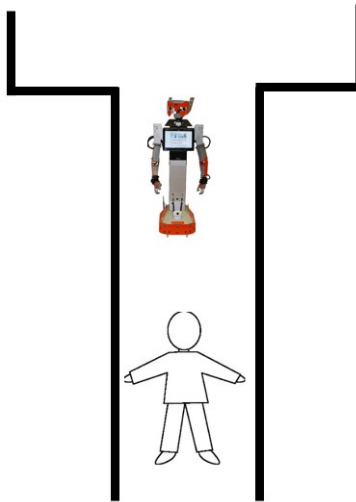


Figure 6.5: The bottleneck problem when the robot and human are moving in narrow places

- Move Right/Left: The user can contribute to select the collision avoidance path to the right or left of him. Fig 6.6 shows the concept of this order. Each

person is represented by the ellipse  $P_i$ , which represents the distance between the right and left shoulders of the person  $i$ .

Supposing that the master P2 asked the robot to move to its left side, the robot will then search for the nearest region that it can go through to the left of the user. It could be seen that the candidate region R2 between the person P3 and P4 is the selected one. The robot will then calculate its path which will be discussed later. After passing the region R2, the robot will give the control to the multi-floor navigation system to allow the robot to keep going to the goal location.

If the master ordered the robot to move to a certain direction, and the robot couldn't find a free candidate region in the selected direction, it will keep stopped and it will send a voice message "no free path" to inform the human that there is no free space. In this case, the master person has to either give the robot another order, or to let the human to move for a certain distance to keep a space for the robot to move, and then the master has to provide again the motion direction to the robot.

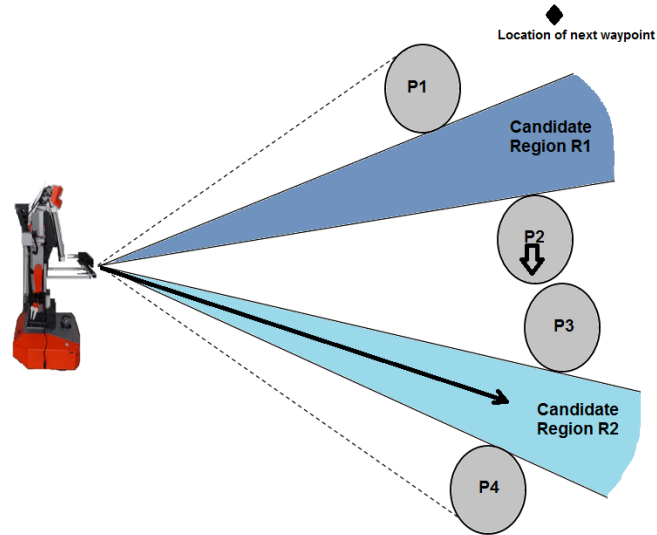


Figure 6.6: Cooperative Collision Avoidance via Interaction

The goal of this interaction is to allow the humans who are working alongside with the robot to supervise it to reach its goal location. Taking into consideration that the worker can have more information regarding the robot's task, the obstacles in the robot's path and the motion directions of the other humans existing in the same location. These circumstances could be faced in many places as laboratories, museums, hospitals, and restaurants.

### 6.3.2. Autonomous Collision Avoidance

If no human interacted with the robot within the given period of time, the robot will consider that the users delegated it to search for the free paths. Fig 6.7 depicts the flow chart of the proposed system.

Basing on the search equations in chapter 6.3, the robot will take into consideration the whole candidate regions that consist a potential collision-free path for the robot. Then, it will get the direction of the next waypoint on the original path of the robot from the multi-floor system, and select the region that is closest to this way-point. In case that the robot couldn't find any free region, it will send the voice message "no free path" to inform the humans that they have to keep a space for it to move. If the human moved allowing enough free space for the robot, it will generate the collision avoidance path and keep moving to the goal location after finishing the path.

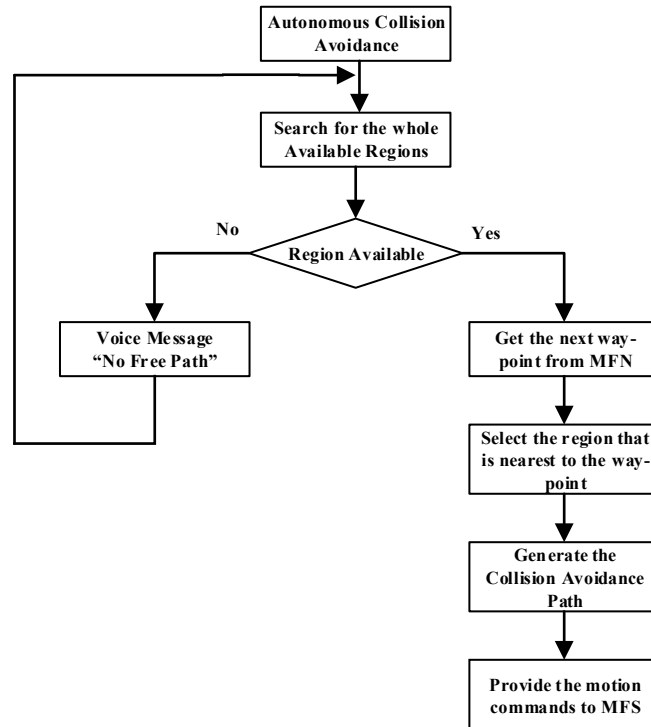


Figure 6.7: The flow chart of Autonomous Collision Avoidance

Fig 6.8 shows four humans located near the robot. The dashed lines represent the horizontal field of view for the Kinect 2.0 sensor which is equal to  $70^\circ$ . It can be seen that two regions are detected for four humans. Since the robot has the freedom in selecting the region, it will select R1 as candidate region since it is closest to the original global path of the robot toward the goal location.

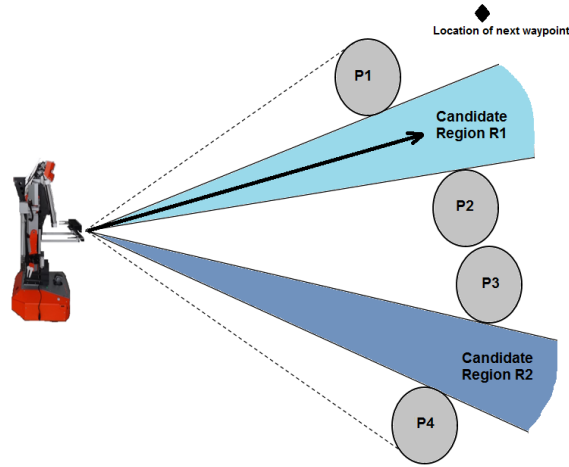


Figure 6.8: Autonomous search for the collision-free path

#### 6.4 Collision Avoidance Path Calculation

After selecting the candidate region using either the cooperative or autonomous collision avoidance, the robot will start to calculate the collision-free path across the selected region.

The calculation of the collision-free path is based on whether the path will be between two humans, or it will avoid a single person. Fig 6.9 shows the calculation of the collision-avoidance path between two humans. To get the path the robot has to move, the following equations are used:

$$x_{ca} = \frac{x_p^{i+1} - x_p^i}{2} \quad (6.11)$$

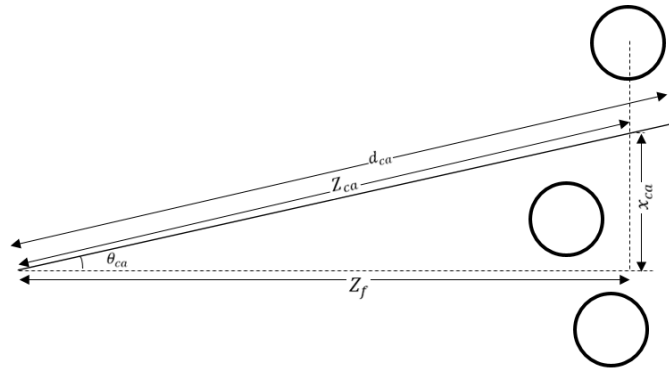


Figure 6.9: Collision-free path calculation between two humans

$$Z_f = \text{Max}\{Z_i, Z_{i+1}\} \quad (6.12)$$

$$\theta_{ca} = \tan^{-1} \frac{x_{ca}}{Z_f} \quad (6.13)$$

$$Z_{ca} = \frac{x_{ca}}{\sin \theta_{ca}} \quad (6.14)$$

$$d_{ca} = Z_{ca} + r_r + r_f \quad (6.15)$$

Where:

$x_{ca}$ : The middle of the selected region.

$Z_f$ : The distance of the farthest person from the robot in the selected region.

$\theta_{ca}$ : The robot's orientation toward the region.

$Z_{ca}$ : The distance between the robot and the middle of the selected region

Fig 6.10 shows the path calculation for the terminal regions. For these regions, the robot uses the following equations for the rightest person:

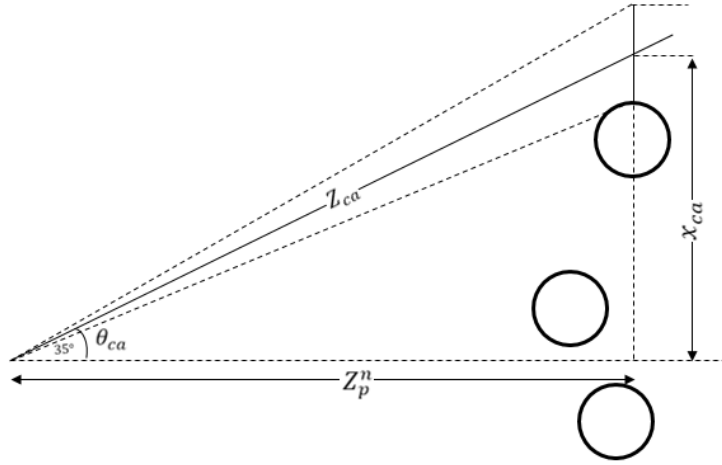


Figure 6.10: The generation of CA path for the terminal person

$$x_{ca} = X_p^n + r_p^n + r_r + d/2 \quad (6.16)$$

$$\theta_{ca} = \tan^{-1} \frac{x_{ca}}{Z_p^n} \quad (6.17)$$

$$Z_{ca} = \frac{x_{ca}}{\sin \theta_{ca}} \quad (6.18)$$

$$d_{ca} = Z_{ca} + r_p^n + r_r \quad (6.19)$$

Similarly, to calculate the terminal region for the most left person in the group, the following equations are used:

$$x_{ca} = X_p^0 - r_p^0 - r_r - d/2 \quad (6.20)$$

$$\theta_{ca} = \tan^{-1} \frac{x_{ca}}{Z_p^n} \quad (6.21)$$

$$Z_{ca} = \frac{x_{ca}}{\sin \theta_{ca}} \quad (6.22)$$

$$d_{ca} = Z_{ca} + r_p^0 + r_r \quad (6.23)$$

Moreover, the same terminal region equations are used when there is only a single person in the path of the robot with taking into consideration that  $X_p^n = X_p^0$ .

## 6.5 Robot's Linear and Angular Velocities Calculation

In collision avoidance, the robot has to adopt its linear and angular velocities based on the risk ratio of the followed collision-free path. Thus, it has to decrease its linear and angular velocities when the path is narrow in cluttered environment, and vice versa.

In the implemented velocity controller, the robot will tune its linear and angular velocities based on the width of the region that it passes. This direct proportion will lead the robot to decrease its linear and angular velocities when the region is narrow, and increase it when the region is wider. To do this, the following equations are used to get the velocities of the robot:

$$v = v_{max} \cdot \frac{\log_{10} \theta_r}{\log_{10} \theta_{max}} \quad (6.24)$$

$$\omega = \omega_{max} \cdot \frac{\log_{10} \theta_r}{\log_{10} \theta_{max}} \quad (6.25)$$

Where  $\theta_{max}$  the maximum vision angle of the Kinect sensor and equal  $70^\circ$ ;  $\theta_r$  the angular width between two humans for middle regions, and between the terminal human and the vision limit for the sensor ( $\pm 35^\circ$ ) for terminal regions.



Fig 6.11 shows the simulation result of the linear and angular velocities of the robot for different angles, given a maximum linear and angular velocities as  $v_{max} = 0.3 \text{ m/s}$  and  $\omega_{max} = 0.5 \text{ rad/s}$ .

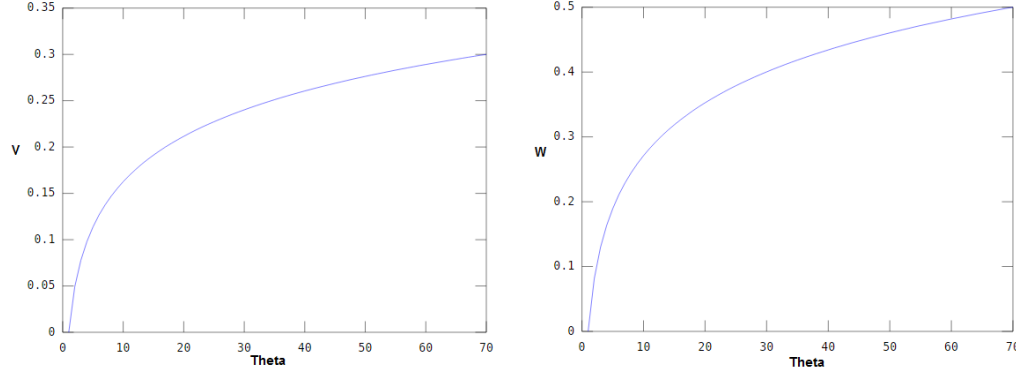


Figure 6.11: The linear and angular velocities for different width of regions

## 6.6 Software Implementation

### 6.6.1 Development Tools

The whole software implementation is realized using C# language which is an object oriented programming language appeared in 2000. C# could be used to implement a wide range of software applications which run on .Net framework [137]. C# is built over the previous C, C++ languages which make it more robust and more convenient for developing different applications including embedded systems and human-machine interfaces.

Moreover, the Extensible Markup Language (XML) is used for data exchange between the collision avoidance system and the multi-floor system. XML is a supporting language which is used for encoding documents and exchange the data between programs. This language uses a set of instructions and formats which are easy to understand by both the human and machines.

### 6.6.2 System Realization

The key success of any engineering system is the integration and synchronization between its components. In mobile robotics, several tasks have to run

simultaneously as collision avoidance, path planning, arm grasping, and charging, to realize an effective autonomous manipulation system.

The overall goal of the research is to build an autonomous transportation system which is able to transport the labware between different laboratories. To realize this, several partial systems have to run in parallel, and integrate with each other to guarantee the success of transportation tasks. Fig. 6.12 shows the general architecture for the implemented mobile transportation system.

- The robot remote center (RRC) is responsible on connecting with the process management system, and sending transportation commands to each robot based on the charging status of the robot and its nearness from the destination [110].
- The motion power control is a software which is located on the PC-Laptop and it is responsible of getting the sensors' information of the robot and controlling the actuators.

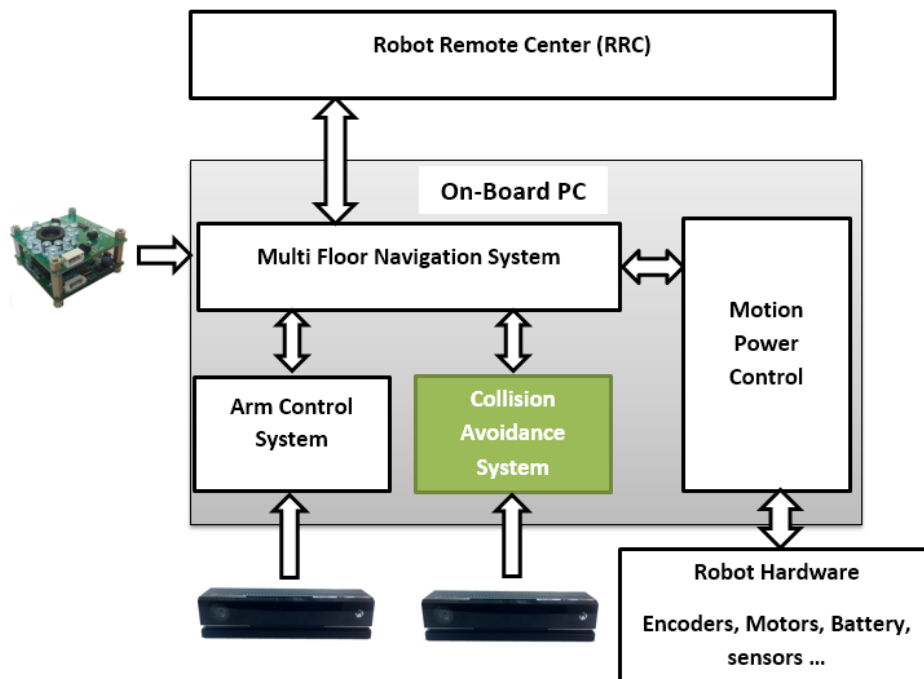


Figure 6.12: The general architecture of the transportation system

- The multi-floor navigation system provides the robot with the robot's location within the global map and guides it to the grasping and charging locations [130].

- The arm control system is responsible on controlling the robot's arms to grasp and place the labware [13].
- The collision avoidance system is a local path planner, which allows the robot to adjust its path and avoid the unexpected obstacles which are located in its path to the goal location [112].

The collision avoidance system will receive the data from the Kinect sensor, and use the algorithms which are discussed before to allow the robot from re-planning its path to avoid the obstacles. Since the proposed collision avoidance system is related to the HRI system, a robust combination must be implemented to synchronize the work of these systems. Moreover, the collision avoidance system must be able to exchange the data with the multi-floor navigation system to get the next way-points and send the motion orders to the robot. Fig 6.13 shows the block box of the implemented collision avoidance system.

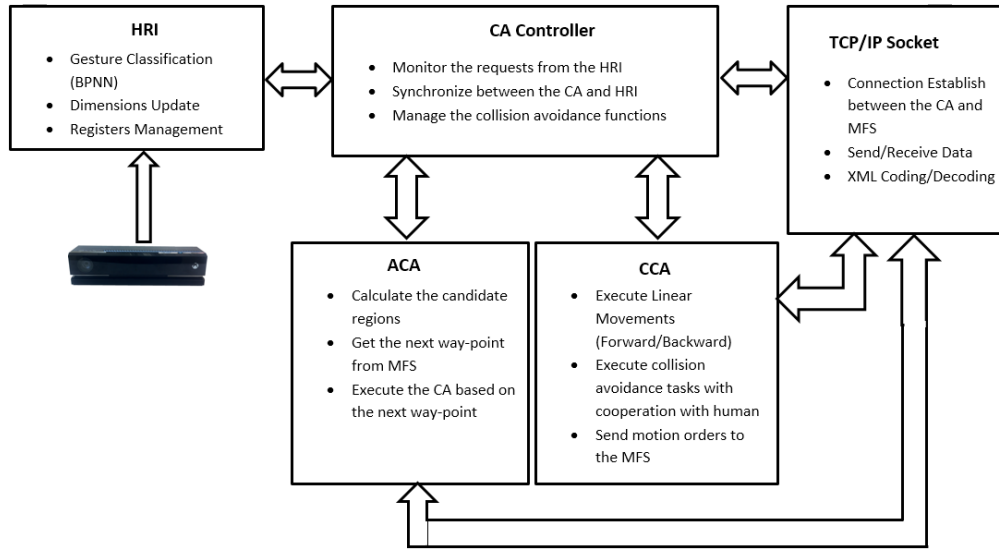


Figure 6.13: The general architecture of the Cooperative Collision Avoidance System

To show the practical implementation of the collision avoidance system, the complete code for the “autonomous collision avoidance” is copied and could be found in appendix 5. The other collision avoidance functions such as “move right/left” have similar concepts of the autonomous collision avoidance.

### 6.6.2.1 Collision Avoidance Controller

The collision avoidance controller is the core of the collision avoidance system. It analysis the information received by the HRI system, monitors the connection state of the client (MFS), and trigs the collision avoidance functions based on the received information from the HRI. Fig 6.14 shows the flow chart of this controller.

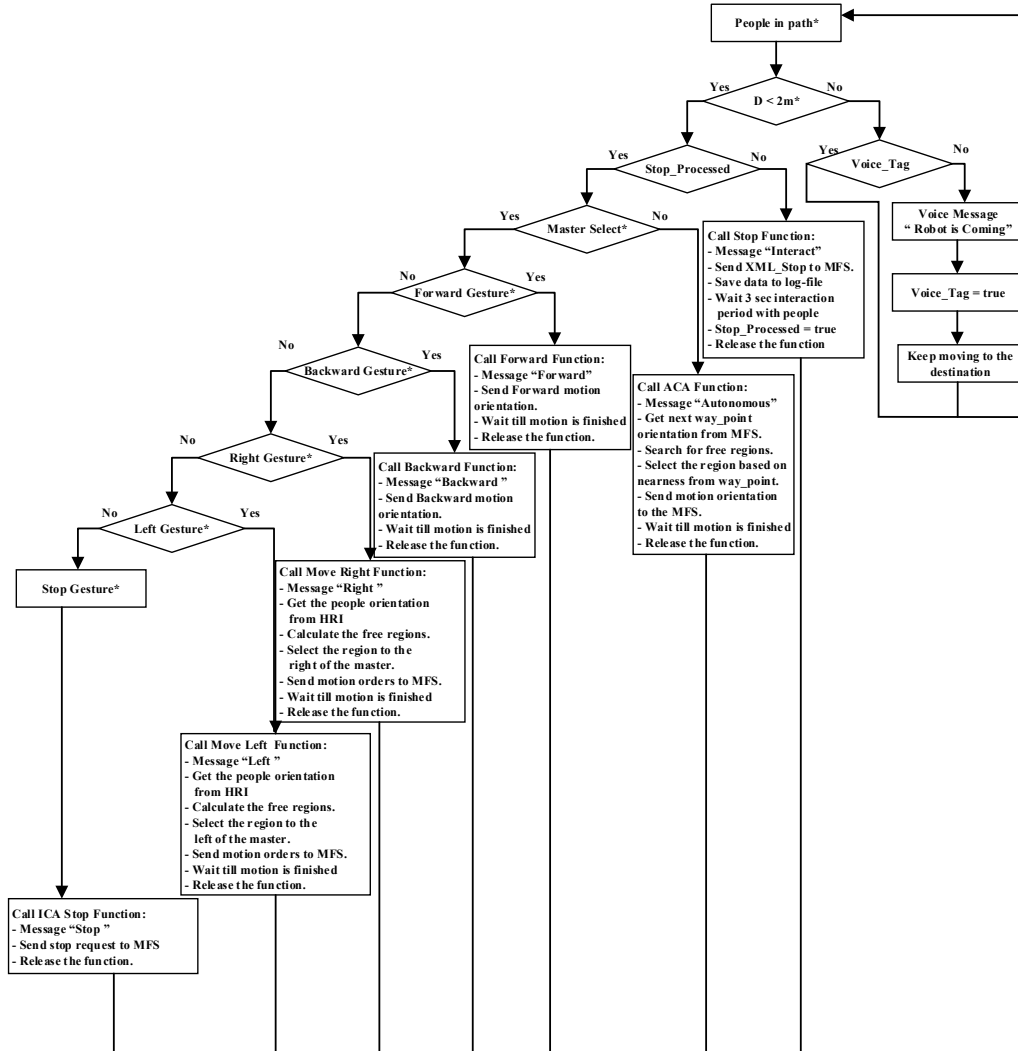


Figure 6.14: The flow chart of collision avoidance controller, (\*) means information received from the HRI system

When a new process cycle starts, the controller checks whether there are humans in the path, and one of them at least has a distance less than 2 m. The controller will activate the “Stop” function which will let the robot stops for three seconds, giving a time distance for the human to interact. The “stop” function will be released and the

control will be given back to the CA controller. If the HRI shows that no human interacted with the robot, the controller will trig the autonomous collision avoidance procedure, or it will trig one of the cooperative collision-avoidance functions if a person interacted with the robot. Each function will implement the required processes, and send the motion commands to the MFS via the socket.

### 6.6.2.2. Communication with the MFS

For exchanging the data between the collision avoidance system and the MFS, a well-structured API is designed based on TCP/IP protocol [112] [130]. Furthermore, the data is transferred using XML format. Each message is composed of a header, and the message body, the header is composed of 4 bytes, and they save the message length in bytes, while the message body is composed of the XML message coded using UTF-8 format.

The collision avoidance system represents the server part of the socket, while the MFS represents the client. The server will keep listening to the socket till a start message “keepOnline” and “StartCA” are received from the client. After each message, the server will reply the client confirming receiving the messages and establishing the connection correctly. The collision avoidance will then start monitoring the existence of humans in the path. When an obstacle is less than 2m, the CCA system will send “ObstacleDetected”, and the MFS will stop the robot and reply by providing the orientations of the next way-point as it could be seen in Fig 6.15. The CCA will then calculate the collision-free path, and it will provide the motion orders to the MFS, by sending the angular and linear distances, and the velocities that the robot has to move. When the path is free of obstacles, the collision avoidance system will send “ObstacleFree” allowing the robot to keep moving to the

```

Connection Establishment
MFS: <MFS-Commands ID="0" Client="MFS-CA"><Transport CommandName="KeepOnline" /></MFS-Commands>
CA: <H20M-Commands ID="0" Client="H20-RBC-V2"><Transport CommandName="KeepOnline" /></H20M-Commands>
MFS: <MFS-Commands ID="0" Client="MFS-CA"><Transport CommandName="StartCA" /></MFS-Commands>
CA: <H20M-Commands ID="0" Client="H20-RBC-V2"><Transport CommandName="StartCA" /></H20M-Commands>

Collision Avoidance Data Transmission
CA: <H20M-Commands ID="1001" Client="H20-RBC-V2"><Transport CommandName="ObstacleDetected" /></H20M-Commands>
MFS: <MFS-Commands ID="1001" Client="MFS-CA"><Transport CommandName="StartAvoidance" XCurrent="16.89"
YCurrent="-0.56" WP_X="17" WP_Y="0.82" /></MFS-Commands>
CA: <H20M-Commands ID="1002" Client="H20-RBC-V2"><Transport CommandName="CAParameters"><KS Distance="0"
CADistance="0" CAAngle="20" CADistanceTime="0" CARotationTime="3981" /></Transport></H20M-Commands>
MFS: <MFS-Commands ID="1002" Client="MFS-CA"><Transport CommandName="CAParameters" EstimateTime="3981" /></MFS-Commands>
CA: <H20M-Commands ID="1003" Client="H20-RBC-V2"><Transport CommandName="CAParameters"><KS Distance="2"
CADistance="2.1" CAAngle="0" CADistanceTime="8414" CARotationTime="0" /></Transport></H20M-Commands>
MFS: <MFS-Commands ID="1003" Client="MFS-CA"><Transport CommandName="CAParameters" EstimateTime="8414" /></MFS-Commands>
CA: <H20M-Commands ID="1004" Client="H20-RBC-V2"><Transport CommandName="ObstacleFree" /></H20M-Commands>

```

Figure 6.15: The XML messages between CA and MFS

goal location.

### 6.6.2.3 Collision Avoidance User Interface

Fig 6.16 shows the user interface for the collision avoidance and human robot interaction systems. The interface is implemented as simple as possible to allow the user from handling the collision avoidance system easily. The upper part shows the connection status with the MFS, the middle part shows the human-robot interaction parameters, and the lower part shows the collision avoidance parameters.

The user interface allows the user from adjusting the parameters of the collision avoidance system, such as adjusting the distance the robot has to stop to implement the collision avoidance, the maximum velocities for any collision-avoidance functions. Furthermore, the user can activate/deactivate the velocity controller, and activate/deactivate the cooperative collision avoidance.

Figure 6.16: The interface of the collision avoidance system

## 6.7 Experimental Results

To check the performance of the collision avoidance system, several experiments have been implemented, and the whole parameters of the system are recorded and analysed. The tests have been implemented on the cooperative collision avoidance functions “Move forward, move backward, move right, move left”, besides to the tests over the autonomous collision avoidance. The whole experimental results are collected in the appendix.

### 6.7.1 Tests over the Cooperative Collision Avoidance

As it is shown before, the cooperative collision avoidance is based on the interaction between the robot and the master person. Furthermore, it is shown that there are four motion orders which could be provided to the robot via interaction: Move forward/backward, Move right/left. Each of these functions is tested and analysed separately.

#### 6.7.1.1 Move Forward

To test the move forward function, 30 experiments have been done on different velocities of the robot as it could be seen in the table “A6.A” in the appendix 6. In each experiment, the robot is asked to move 1.2m as a result of the interaction. Fig 6.17 shows an experiment for the “move forward” test. A person raises his right arm 180° to inform the robot that he will be the master, then the master orders the robot

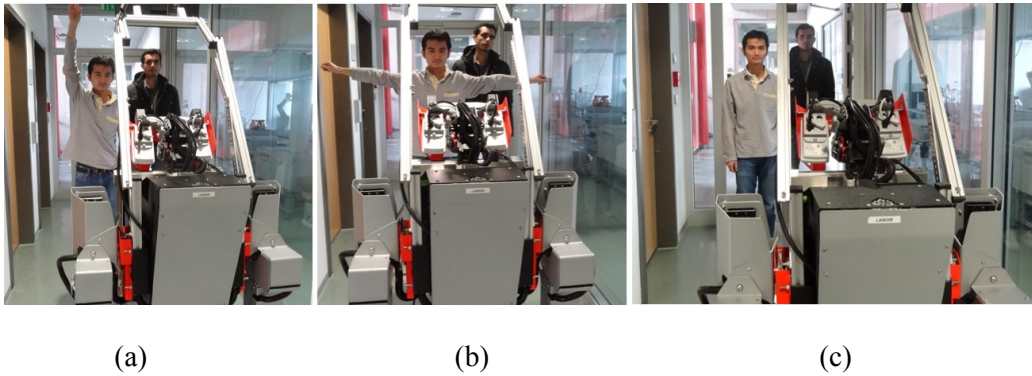


Figure 6.17: The experiment for “move forward” function (a) the user raises “master” gesture, (b) orders the robot to move forward, (c) the robot executes the order

to move forward to guide it to another location, since the robot and the people are met in narrow corridor so neither of them is able to avoid each other. Table 6.1 summarizes these experiments.

Table 6.1: Summary of the experiments for “move forward” function

V(m/s)	No. Experiments	Time(sec)	Note
0.1	10	12	
0.2	10	6	
0.3	10	4	Deviation $[2^\circ, 4^\circ]$

The experiments show that the robot successfully interacted with the master, and executed the “forward” order. Furthermore, it could be seen that the robot on higher velocities (0.3 m/s) doesn’t follow the straight path, and it deviates around  $[2^\circ, 4^\circ]$  from the straight path due to problems in the wheels of the robot.

### 6.7.1.2 Move Backward

Similarly to “move forward”, 30 experiments were implemented to test the “move backward” function for the robot. Likewise the “move forward”, this function could be used to solve the bottleneck problem, and guide the robot to another wide area so the people and the robot can avoid each other. Fig 6.18 shows an experiment for this function, a person raises his right arm  $180^\circ$  to allow the robot from interacting with him, then the master moves his both arms  $180^\circ$  to order the robot to move backward.

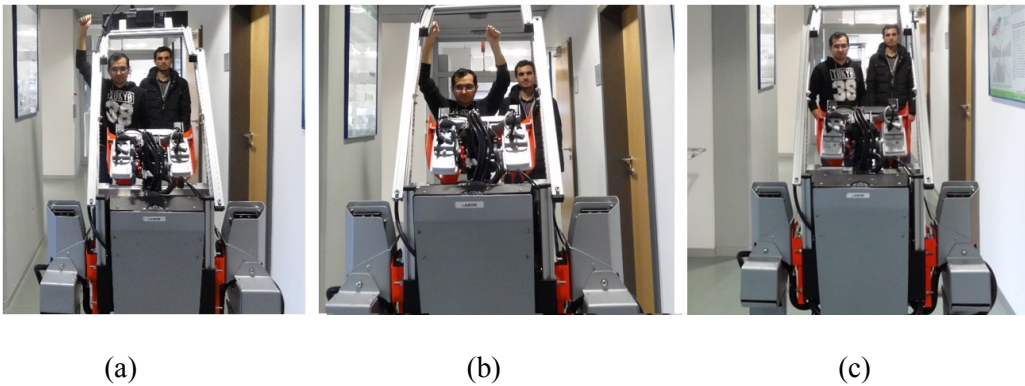


Figure 6.18: The experiment for “move backward” function (a) the user raises “master” gesture, (b) orders the robot to move backward, (c) the robot executes the order



Table “A6.B” in appendix 6 shows the experimental results for this function for three velocities. While the table 6.2 summarizes these experiments.

Table 6.2: Summary of the experiments for “move backward” function

V(m/s)	No. Experiments	Time(sec)	Note
0.1	10	12	
0.2	10	6	
0.3	10	4	Deviation $[2^\circ, 4^\circ]$

It can be seen from the table 6.2 that at velocity (0.3 m/s), the robot shows a deviation from the straight path, due to the low accuracy for the robot’s wheels.

### 6.7.1.3 Move Right

In “move right” function, the robot will interact with the master, and consider only the regions that are located to the right of him. To test the function, 30 experiments were implemented and divided into three groups. Each group includes 10 experiments with a certain maximum linear velocity  $v_{max}$  and angular velocity  $\omega_{max}$ . Furthermore, it could be seen that in each group, the robot moved in the middle of two people for 5 experiments, and to the right of the last terminal human for the other 5 experiments. Tables “A6.C, A6.D, A6.E” in the appendix 6 show the experimental results for these experiments, while table 6.3 summarizes the experiments. Furthermore, Fig 6.19 shows the real test environment for the robot.

Table 6.3: The experimental summary for the “Move Right” function

Table	Number of Experiments	V(m/s)	W(rad/s)	Note
A6.C	(10) 5 right, 5 middle	0.2	0.4	1 missed skeleton
A6.D	(10) 5 right, 5 middle	0.25	0.3	1 missed skeleton
A6.E	(10) 5 right, 5 middle	0.3	0.5	3 deviations 1 missed skeleton

In each experiment, it could be seen that the robot searches for the whole regions which are located to the right of the master, and they are wide enough to generate the collision-avoidance path. The available regions could be found in the tables under “region candidate”. After selecting the region, the robot will calculate its

width  $\theta_r$  to compute the linear velocity ( $V$ ), and angular velocity ( $\omega$ ) of the robot. Finally, the robot calculates the heading angle and the distance “d” of the collision avoidance path that it has to move to pass the human.

“missed skeleton” in the experiments means that when the robot stops for interaction, the Kinect took a time to detect the people, and this required from them to shake their bodies a bit to allow the Kinect from detecting them. While the deviation means that the robot didn’t move accurately to the planned path, because of the limitation in the wheels and the motors of the robot.

In summary, the function was able accurately to select the whole available regions, calculate the linear and angular velocities of the robot based on the angular width of the selected region, and pass the people without causing collision with them.

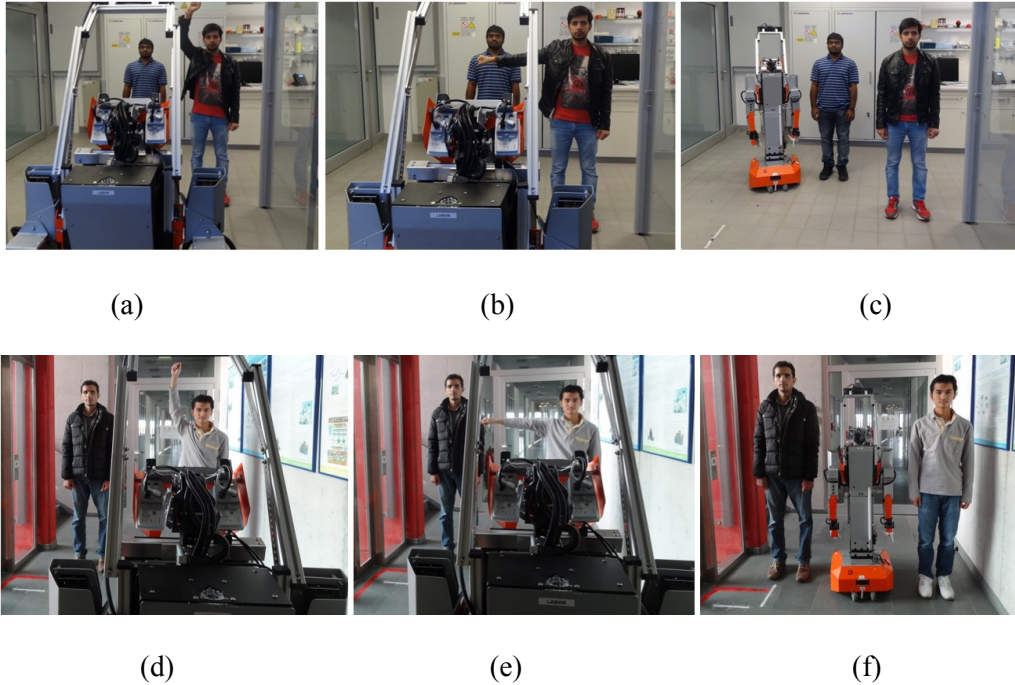


Figure 6.19: The experiments for the “Move Right” function (a, d) the master person is selected, (b, e) the master orders the robot to move to the right, (c) the robot selects the terminal region to the right of the last person, (f) the robot selects the middle region between the master and the other person

### 6.7.1.4 Move Left

Similarly to the tests over the “move right” function, the same experimental procedures also followed to test the “move left” function. The 30 experiments were divided into three groups. In each group the robot is tested under different maximum linear and angular collision-avoidance velocities. Fig 6.18 shows the robot’s movement between two people, and to the left of the terminal human. Furthermore, the calculated candidate regions, region width, and the velocities could be found in the tables “A6.F, A6.G, A6.H” in appendix 6, and the total experiments were summarized in the table 6.4.

It could be concluded that out of 30 experiments, there were three times the human required to shake a bit to allow the Kinect from recognizing the skeletons. Furthermore, it is shown that at the higher velocity ( $v_{max} = 0.3 \text{ m/s}$ ) the robot deviates for around  $3^\circ$  to  $4^\circ$  from its planned path. Still, in the whole experiments the collision avoidance system was able to calculate the whole parameters correctly, and avoid the people which are located in the path.

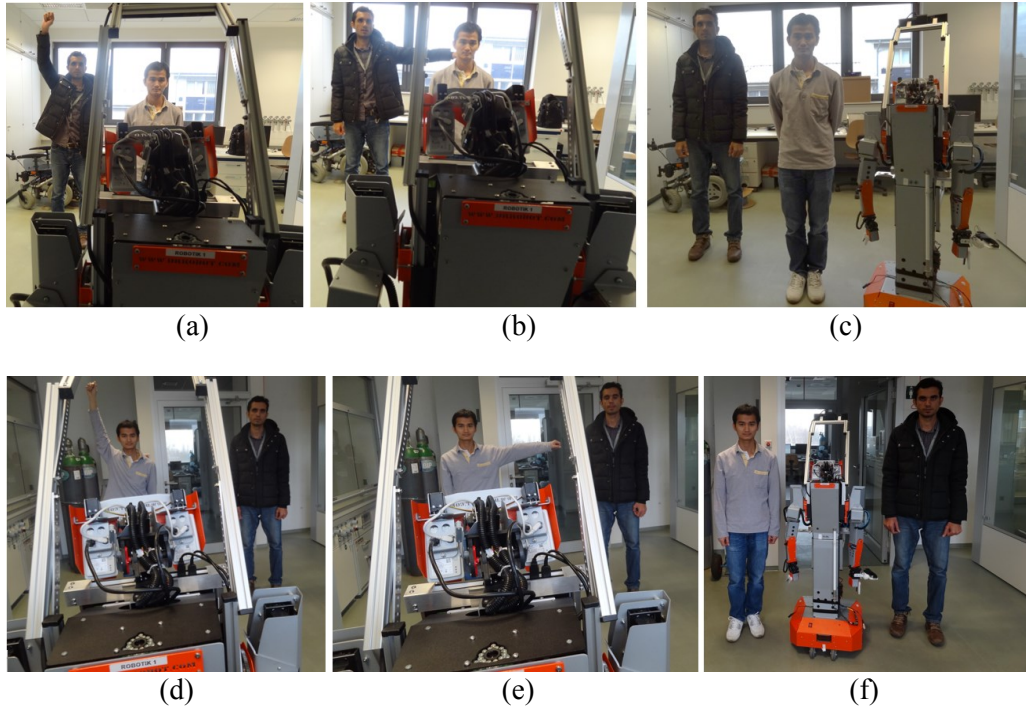


Figure 6.20: The experiments for the “Move left” function (a, d) the master person is selected, (b, e) the master orders the robot to move to the left, (c) the robot selects the terminal region to the left of the last person, (f) the robot selects the middle region between the master and the other person

Table 6.4: The experimental summary for the “Move Left” function

Table	Number of Experiments	V(m/s)	W(rad/s)	Note
A6.F	(10) 5 left, 5 middle	0.15	0.3	1 missed skeleton
A6.G	(10) 5 left, 5 middle	0.2	0.4	1 missed skeleton
A6.H	(10) 5 left, 5 middle	0.3	0.25	2 deviations 1 missed skeleton

### 6.7.2 Autonomous Collision Avoidance (ACA)

In autonomous collision avoidance, the robot will implement the collision avoidance path autonomously without interaction with the human. The robot will search for the whole available regions, and then it will select the region that is closest to its original path, by comparing the angle of the next waypoint and the heading angle of each candidate region. Fig 6.21 shows two experiments for the autonomous collision avoidance. The robot in (a, b, c) could detect two free regions, and it will select the

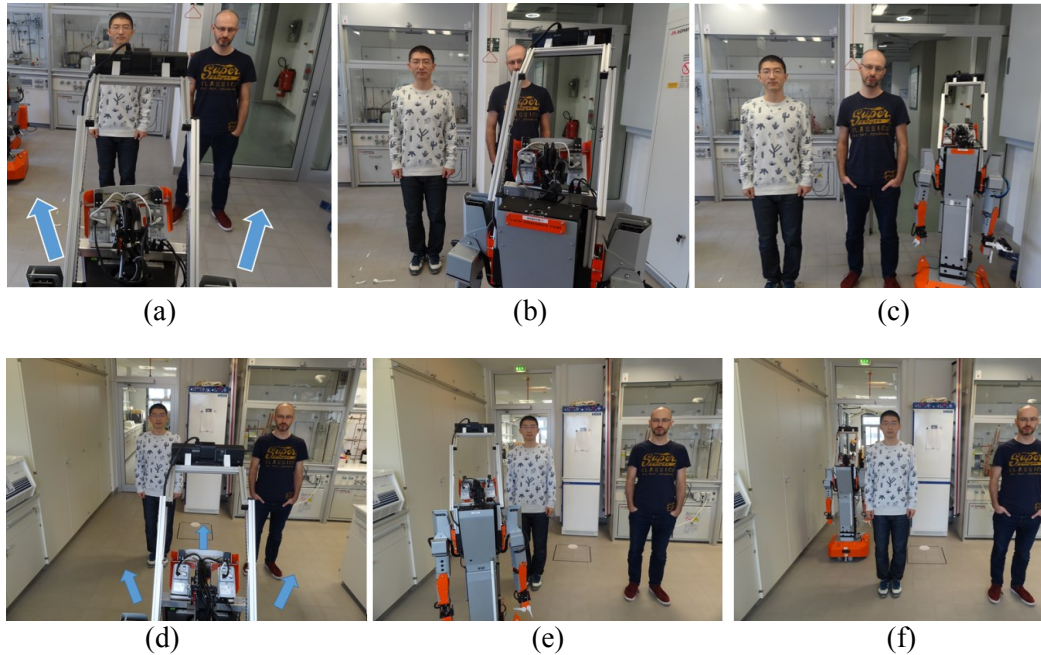


Figure 6.21: The experiments for “autonomous collision avoidance”, (a) The robot found two free regions, (d) the robot could find three free regions, (b, d) the robot selects the region that is closest to the next way-point, (c, f) the robot calculates the collision-free path and avoid the people

region to the right since it is nearest to its original path. In Fig 6.21 (d, e, f) the robot detected three regions, and it selected the region that is to the left since it is nearest to the next waypoint.

Tables “A6.I, A6.J, A6.K” in appendix 6 show the experimental results of the ACA. While table 6.5 summarizes the total experiments.

Table 6.5: The experimental summary for the Autonomous Collision Avoidance

Table	Number of Experiments	V(m/s)	W(rad/s)	Note
A6.I	10	0.15	0.2	1 missed skeleton
A6.J	10	0.2	0.5	1 missed skeleton
A6.K	10	0.3	0.25	2 Deviations 1 missed skeleton

In the tables, the “way-point” blank represents the angle that the robot has to move to reach the next way-point on its original path to the goal location. It could be seen then that when there is more than one candidate region, the robot will select the region that is nearest to the angle of next way-point. The robot will then use the width of the selected region  $\theta_r$  to calculate the robot’s velocities. Finally, the robot calculates the heading angle and the travelled distance to pass the people.

From table 6.5, it could be found that the people required to shake a bit to be recognized from the Kinect in three experiments. Furthermore, the robot deviated for around  $4^\circ$  in two experiments at the higher velocity of the robot.

The experiments show that the autonomous collision avoidance system works with a success rate 100%, and it can correctly select the region that is closest to the next waypoint, and provide the collision-free paths to the robot.

### 6.7.3 Test of the Collision Avoidance for different Situations

To check the performance of the collision avoidance system, two additional experiments have been implemented. In each experiment, the robot is moved in a certain path where humans are existed. Furthermore, the collision avoidance parameters are recorded to check the system.

In the first experiment, the robot will move from the charging station, to a laboratory to pick a sample. In Fig 6.22, the blue line represents the original path of the robot that it had to follow when no people are existed in the path, while the red line represents the adjusted path of the robot after avoiding the people. The maximum velocities are set to  $v_{max} = 0.25 \text{ m/s}$ ,  $\omega_{max} = 0.4 \text{ rad/s}$ . The total time the robot needed to cross the path without generating collision avoidance paths is 207s. Fig 6.22 shows the experiment in the three locations.

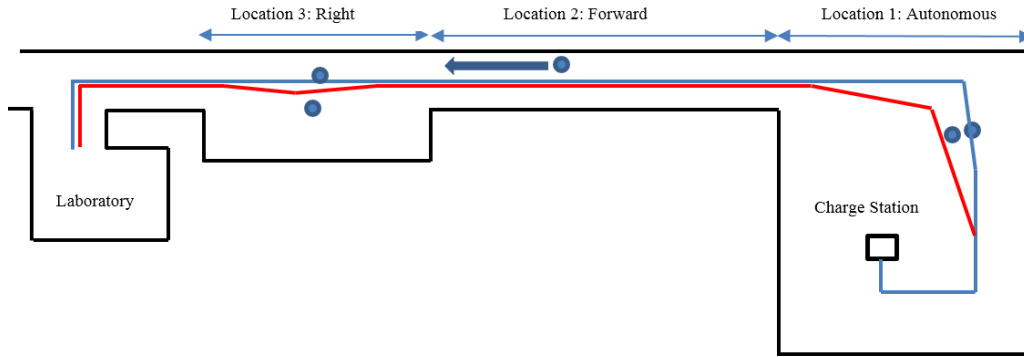


Figure 6.22: The followed path in the first experiment

In location 1, the robot met two humans, and no one interacted with it. Thus, the robot implemented the collision avoidance autonomously. The robot selected the right region since it is nearest to its original path.

In location 2, the robot met humans in the corridor, since there is no free space for implementing the collision avoidance, the master raised his two arms horizontally to order the robot to move forward, till it reached to the location 3.

In location 3, the master person asked the robot to move to the right since there is enough space for both the robot and the human to avoid each other. The robot will then execute the cooperative collision avoidance by searching for the region that is located to the right of the master person. Fig 6.23 shows the motion of the robot in the three locations. While Table 6.6 summarizes the three experiments. The total time needed to implement the experiment is 264s.



Table 6.6: The experimental results for the first path

	Location 1	Location 2	Location 3
CA Function	ACA	Move Forward	CA Move Right
Region 1 (deg)	-31.4	-----	9.9
Region 2 (deg)	19.5	-----	34.5
Waypoint (deg)	8.4	-----	-----
$\theta$ (deg)	19.5	0	9.9
$d$ (m)	3.28	1.2	3.14
$\theta_r$ (deg)	27.9	-----	31.1
$v$ (m/s)	0.195	0.25	0.202
$\omega$ (rad/s)	0.31	0	0.323
Time(ms)	17863	4800	16060

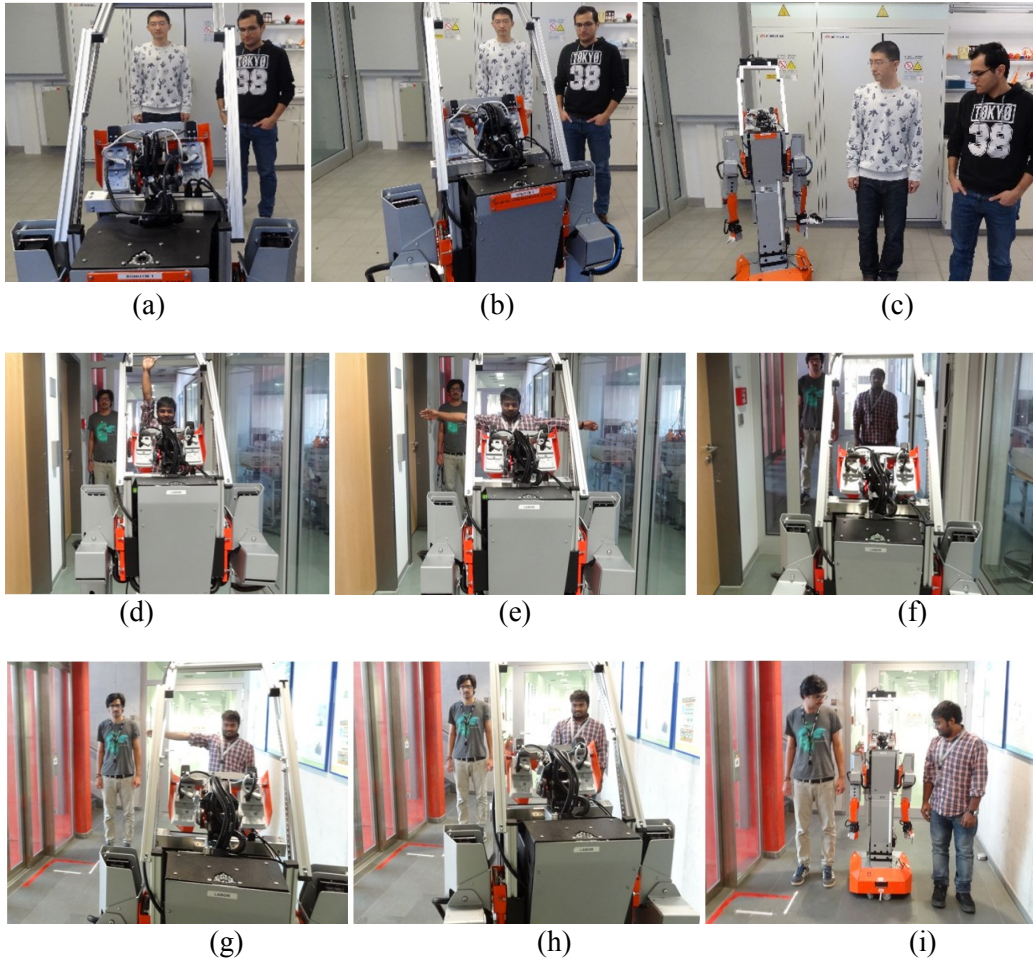


Figure 6.23: The motion of the robot for three locations. (a, b, c) no people interacted with the robot, so it implemented the collision avoidance autonomously. (d, e, f) the robot met people in narrow path, so the master ordered the robot to move forward till the end of the corridor. (g, h, i) the master asked the robot to move to the right

In the second experiment, the robot is asked to move in the path which is shown in Fig 6.24 as a blue dashed line. The required time to pass the path without collision avoidance paths is 94 seconds. The maximum linear and angular velocities are set to  $v_{max} = 0.2 \text{ m/s}$ ,  $\omega_{max} = 0.3 \text{ rad/s}$ .

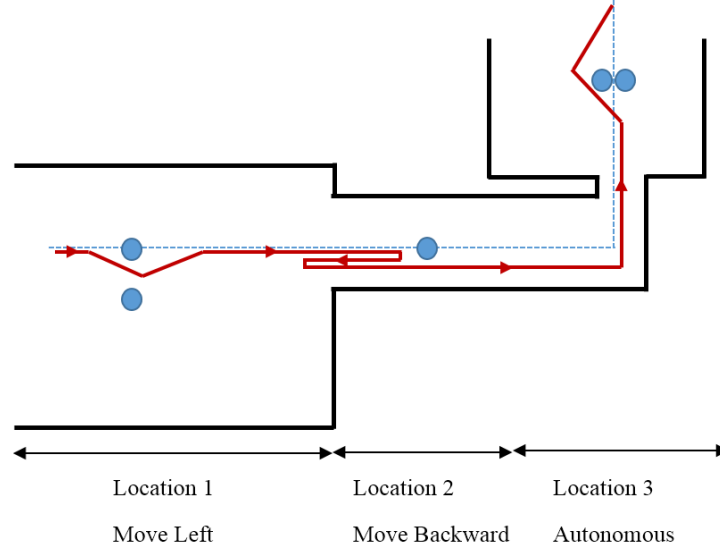


Figure 6.24: The followed path in the second experiment

In the first location, the human is interacted with the robot and he asked him to move to the left. In location 2, the robot is moved in narrow path, so the human interacted with the robot and asked it to move backward. In the third location, the robot faced two people in the path, and it implemented autonomous collision avoidance since none of the people interacted with it. Table 6.7 shows the

Table 6.7: The experimental results for the second path

	Location 1	Location 2	Location 3
CA Function	CCA Move Left	Move Backward	ACA
Region 1 (deg)	-22.3	-----	-31.2
Region 2 (deg)	11.1	-----	14.6
Waypoint (deg)	-----	-----	-25.0
$\theta$ (deg)	11.1	-----	-31.2
$d$ (m)	2.81	1.2	2.90
$\theta_r$ (deg)	36.0	-----	17.9
$v$ (m/s)	0.168	0.2	0.135
$\omega$ (rad/s)	0.252	0	0.202
Time(ms)	17438	6000	24092



experimental results of the complete experiment, while Fig 6.25 shows the motion of the robot in the real environment. The robot successfully implemented the three motions, and it required 156s to reach the goal.

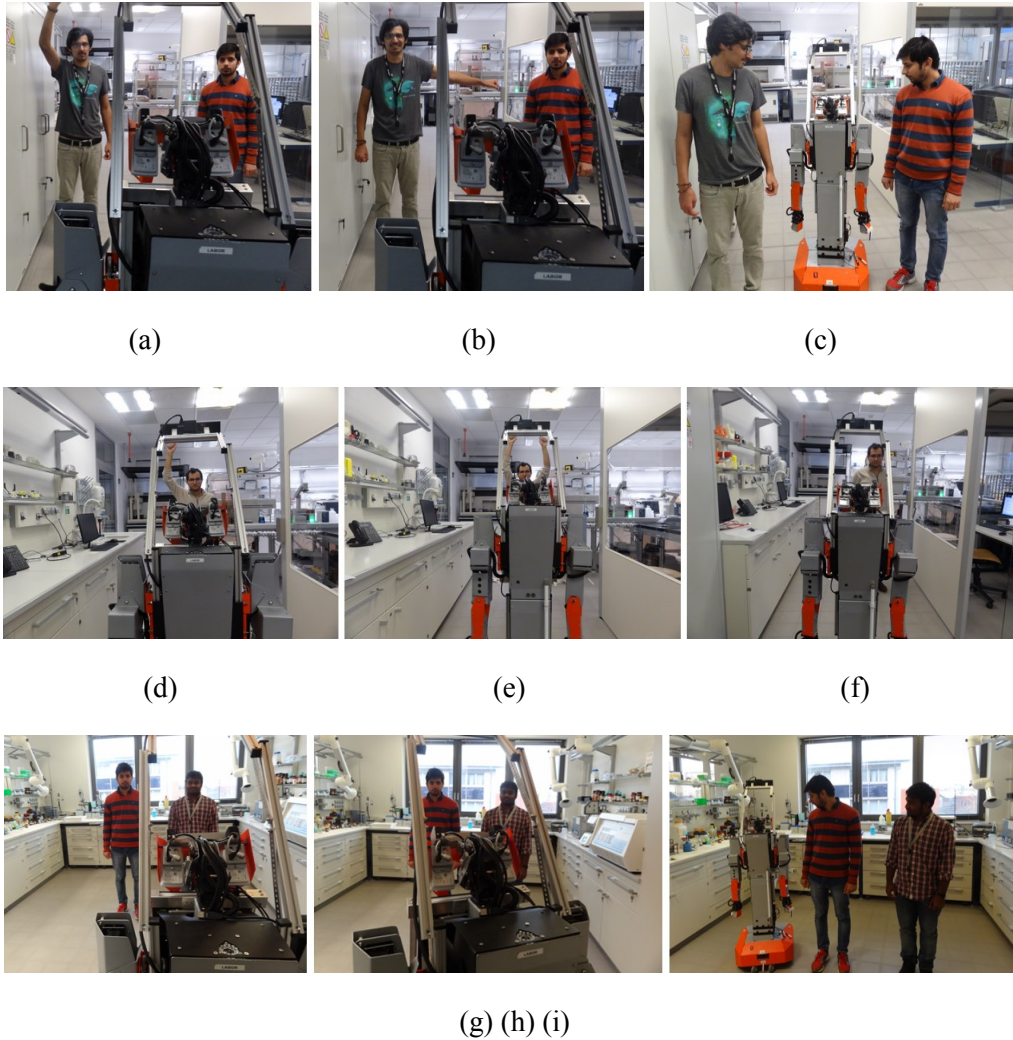


Figure 6.25: The motion of the robot in the three locations for the second experiment.

(a, b, c) the master asked the robot to move to left, and the robot select the middle region, (d, e, f) the master ordered the robot to move backward since there is no wide path, (g, h, i) no people interacted with the robot, so it implemented autonomous collision avoidance

## 6.8 Discussion

Out of the previous experiments, it could be concluded the following:

- The collision avoidance system together with the human-robot interaction show robust performance; the collision avoidance was able to detect the whole available regions, and select the regions based on the collision avoidance function “autonomous or cooperative”.
- Furthermore, the collision avoidance system implemented the whole calculations accurately (path distances and angles, robot’s velocities), and this provided a smooth movement for the robot between the humans who are located in narrow distances.
- In comparison with the other collision avoidance systems, the proposed system has the advantage of allowing the people from controlling it in some situations such as the existence of several people in the path, or when the user knows that the robot is moving toward a cluttered path, so the master can ask the robot to move to a certain direction where there is no obstacles. Furthermore, the system allows the operator from interrupting the motion of the robot, stop it and control it in emergency situations.
- The previous collision avoidance systems didn’t solve the bottleneck problem, in which the robot and the people meet in narrow paths so neither of them is able to avoid the other. The implemented system solved this problem, by allowing the master human from guiding the robot to another location which is wide enough to avoid each other, and complete their path safely after the interaction.
- On the other hand, it is noticed that due to the vibration of the robot, the Kinect sensor loses some skeleton frames, and this required from the master to raise his arm again to inform the robot to interact with him. Furthermore, it is shown in some cases, when the robot stops, it causes the Kinect to lose the skeleton of people due to the vibration, and it required from them to shake to let the Kinect from detecting them again.

---

## Chapter 7

### Conclusion and Outlook

---

#### 7.1 Conclusion

This dissertation aimed to improve the navigation and the operations of the mobile robots which are navigating in indoor environments alongside to humans. This includes implementing a robust localisation system to enable the robot from defining its location correctly in the work environment; building a human-robot interaction system so the robot can define the human, interact with him via analysing the arms' movements; besides to implement a robust collision avoidance system based on the realized human-robot interaction system to ensure that both the human and the robot will be able to avoid the collisions.

- An improved localization system is implemented for the indoor mobile robots. The StarGazer sensor is used for this task. This sensor has the advantages of using passive landmarks, and the possibility of modifying and extending the map easily to meet the adjustments in the work environment. Still, this sensor suffers from the noises resulted from strong sunlight and fluorescent light. To overcome these noisy measurements, an improved Kalman filter model is applied over the measurements of the sensor. When a new measurement is obtained from the sensor, the filter will use this measurement in the update equation. The filter will then compare the results of the update equation and the estimate equation; if the difference is outer than a certain domain, the filter will consider the result of the estimate equation as the new position of the robot, else it will use the result of the update equation as the new position. The experimental results show that the filter is able to detect the false measurements, and provide estimation for the location of the robot under the false measurements.
- A human-robot interaction system is implemented for the H20 robot to be able to detect the human in the work environment and interact with them. Kinect V2 sensor is a 3D sensor which provides the robot with skeleton frames for up to six humans to the robot. Each skeleton provides the 3D dimensions for 25 joints of the user. Seven gestures are used in this work “master select, move right, move

left, move forward, move backward, resume, and stop”. To analyse and define the gestures, the dimensions of five joints are taken (right and left wrists and elbows, and the neck joint). To classify the gestures, two methods are used and compared: Support Vector Machine and Back Propagation Neural Network. K-fold cross validation training algorithm is used to train and test the models. The experimental results showed a better performance for the BPNN model than the L-SVM model, with the ability to classify 100% of the test data successfully. Furthermore, the real experiments over the human-robot interaction system shows that the system is able to recognize the whole gestures and classify them with a success rate 100%. The implemented human-robot interaction system is integrated with the collision avoidance system.

- A new collision avoidance system is developed for the H20 robot. This system has two levels: cooperative collision avoidance based on human robot interaction, and autonomous collision avoidance. Unlike the majority of traditional collision avoidance systems, the proposed system gives mutual responsibilities for both the human and the robot to avoid each other. In this system, the robot delegates the human to control the collision avoidance procedures by supervising the motion of the robot via interaction. The robot will wait for three seconds allowing the human to interact with it. If a person interacted with the robot, then it will move forward/backward when the user raises the corresponding gestures, or it will execute the collision avoidance path to the right/left of the master person when he points to the right/left using his arms. If no human interacted with the robot, it will then select the free region that is closest to the original direction that the robot was following before the humans appeared in the path. The key advantage of this system is that it gives the human the opportunity to master the motion of the robot when needed, so the robot moves based on the orders provided by the master. The experiments show the ability of the robot from finding the collision-free paths in both situation (autonomous and cooperative), besides to overcome the bottleneck problem by using the orders (move forward, move backward).

## 7.2 Outlook

- The proposed filtration algorithm for the StarGazer sensor can cope the problem of false measurements. In some situations, the measurements of the sensor are disconnected when the sensor changes the landmarks during its movement to the goal location. To overcome such problem, it is possible to implement a sensor fusion between the StarGazer and another sensor such as the encoder, so the filter can provide better estimation for the robot's location.
- Furthermore, it is shown that the Kinect sensor loses some skeleton frames because of the sensor's vibration resulted from the motion of the robot. To solve this problem, the recursive filters such as Kalman filter could be used to compensate the lost frames, and provide estimation for the position of humans to the robot. Another solution is to use the colour camera of the Kinect sensor to detect the human, and implement gesture recognition system based on the data provided by the camera.
- The current collision avoidance system is able to detect only the human. Thus, it is required to implement further work to detect the static obstacles. Fortunately, the 3D vision feature of the Kinect allows from using the same sensor in detecting both the static and human obstacles. After detecting the static obstacles, all what is needed is to provide the positions and widths of these obstacles to the implemented collision avoidance system, so it can take into consideration the distribution of these obstacles when it searches for the free regions.

## REFERENCES

- [1] V. Graefe and R. Bischoff, "Past, present and future of intelligent robots," in *2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003. Proceedings*, 2003, vol. 2, pp. 801–810 vol.2.
- [2] M. T. Mason, "Creation Myths: The Beginnings of Robotics Research," *IEEE Robot. Autom. Mag.*, vol. 19, no. 2, pp. 72–77, Jun. 2012.
- [3] "<http://www.ifr.org/service-robots/statistics/>," *International Federation of Robotics*.
- [4] D. S. Lütjohann, N. Jung, and S. Bräse, "Open source life science automation: Design of experiments and data acquisition via 'dial-a-device,'" *Chemom. Intell. Lab. Syst.*, vol. 144, pp. 100–107, May 2015.
- [5] S. Chowdhury, A. Thakur, P. Svec, C. Wang, W. Losert, and S. K. Gupta, "Automated Manipulation of Biological Cells Using Gripper Formations Controlled By Optical Tweezers," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 338–347, Apr. 2014.
- [6] A. Croxatto, G. Prod'hom, F. Faverjon, Y. Rochais, and G. Greub, "Laboratory automation in clinical bacteriology: what system to choose?," *Clin. Microbiol. Infect.*, vol. 22, no. 3, pp. 217–235, Mar. 2016.
- [7] H. O. Unver, "System Architectures Enabling Reconfigurable Laboratory-Automation Systems," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 41, no. 6, pp. 909–922, Nov. 2011.
- [8] X. Chu, H. Fleischer, N. Stoll, M. Klos, and K. Thürow, "Application of dual-arm robot in biomedical analysis: Sample preparation and transport," in *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, 2015, pp. 500–504.
- [9] A. Allwardt, S. Holzmüller-Laue, C. Wendler, and N. Stoll, "A high parallel reaction system for efficient catalyst research," *Catal. Today*, vol. 137, no. 1, pp. 11–16, Aug. 2008.
- [10] Y. Li, S. Junginger, N. Stoll, and K. Thürow, "4D simulation and Control System for Life Science Automation," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 802–807.
- [11] X. Chu, H. Fleischer, T. Roddelkopf, N. Stoll, M. Klos, and K. Thürow, "A LC-MS integration approach in life science automation: Hardware integration and software integration," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 979–984.
- [12] A. A. Abdulla, H. Liu, N. Stoll, and K. Thürow, "Multi-floor navigation method for mobile robot transportation based on StarGazer sensors in life science automation," in *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, 2015, pp. 428–433.
- [13] M. M. Ali, H. Liu, R. Stoll, and K. Thürow, "Arm grasping for mobile robot transportation using Kinect sensor and kinematic analysis," in *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, 2015, pp. 516–521.
- [14] H. Liu, N. Stoll, S. Junginger, and K. Thürow, "An application of charging management for mobile robot transportation in laboratory environments," in *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*, 2013, pp. 435–439.
- [15] H. Liu, N. Stoll, S. Junginger, and K. Thürow, "A floyd-genetic algorithm based path planning system for mobile robots in laboratory automation," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 1550–1555.

- 
- [16] I. Nishitani, T. Matsumura, M. Ozawa, A. Yoroze, and M. Takahashi, "Human-centered — space path planning for mobile robot in dynamic environments," *Robot. Auton. Syst.*, vol. 66, pp. 18–26, Apr. 2015.
  - [17] J. Palacin, J. A. Salse, I. Valganon, and X. Clua, "Building a mobile robot for a floor-cleaning operation in domestic environments," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 5, pp. 1418–1424, Oct. 2004.
  - [18] Z. M. Bojan Babić, "Towards implementation and autonomous navigation of an intelligent Automated Guided Vehicle in Material Handling Systems," *Iran. J. Sci. Technol. Trans. B Eng.*, vol. 36, no. M1, pp. 25–40, 2012.
  - [19] F. Penizzotto, E. Slawinski, and V. Mut, "Laser Radar Based Autonomous Mobile Robot Guidance System for Olive Groves Navigation," *Lat. Am. Trans. IEEE Rev. IEEE Am. Lat.*, vol. 13, no. 5, pp. 1303–1312, May 2015.
  - [20] A. Cherubini, F. Spindler, and F. Chaumette, "Autonomous Visual Navigation and Laser-Based Moving Obstacle Avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2101–2110, Oct. 2014.
  - [21] C.-C. Hsu, C.-Y. Lai, C. Kanamori, H. Aoyama, and C.-C. Wong, "Localization of mobile robots based on omni-directional ultrasonic sensing," in *2011 Proceedings of SICE Annual Conference (SICE)*, 2011, pp. 1972–1975.
  - [22] S. J. Kim and B. K. Kim, "Dynamic Ultrasonic Hybrid Localization System for Indoor Mobile Robots," *IEEE Trans. Ind. Electron.*, vol. 60, no. 10, pp. 4562–4573, Oct. 2013.
  - [23] L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese, "Filters for mobile robots: EKF, UKF and sensor switching - experimental results," in *2011 9th IEEE International Conference on Control and Automation (ICCA)*, 2011, pp. 925–930.
  - [24] J. Park, H. Lee, Y. Hwang, S. Hwang, and J. Lee, "Beacon scheduling for efficient localization of a mobile robot," in *2011 IEEE International Symposium on Industrial Electronics (ISIE)*, 2011, pp. 870–874.
  - [25] H. Yucel, T. Ozkir, R. Edizkan, and A. Yazici, "Development of indoor positioning system with ultrasonic and infrared signals," in *2012 International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2012, pp. 1–4.
  - [26] A. Yazici, U. Yayan, and H. Yucel, "An ultrasonic based indoor positioning system," in *2011 International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2011, pp. 585–589.
  - [27] Y. Dobrev, S. Flores, and M. Vossiek, "Multi-modal sensor fusion for indoor mobile robot pose estimation," in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016, pp. 553–556.
  - [28] R. Nath, "A TOSSIM based implementation and analysis of collection tree protocol in wireless sensor networks," in *2013 International Conference on Communications and Signal Processing (ICCSP)*, 2013, pp. 484–488.
  - [29] C.-H. Lin and K.-T. Song, "Probability-Based Location Aware Design and On-Demand Robotic Intrusion Detection System," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 44, no. 6, pp. 705–715, Jun. 2014.
  - [30] X. Li, R. Falcon, A. Nayak, and I. Stojmenovic, "Servicing wireless sensor networks by mobile robots," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 147–154, Jul. 2012.
  - [31] T. Alhmiedat, F. Omar, and A. A. Taleb, "A hybrid tracking system for ZigBee WSNs," in *2014 6th International Conference on Computer Science and Information Technology (CSIT)*, 2014, pp. 71–74.
  - [32] L. Yu, Q. Fei, and Q. Geng, "Combining Zigbee and inertial sensors for quadrotor UAV indoor localization," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, 2013, pp. 1912–1916.

- 
- [33] B. Song, G. Tian, G. Li, F. Zhou, and D. Liu, "ZigBee based wireless sensor networks for service robot intelligent space," in *2011 International Conference on Information Science and Technology (ICIST)*, 2011, pp. 834–838.
  - [34] M. L. Rodrigues, L. F. M. Vieira, and M. F. M. Campos, "Mobile Robot Localization in Indoor Environments Using Multiple Wireless Technologies," in *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, 2012, pp. 79–84.
  - [35] M. Barczyk, S. Bonnabel, J.-E. Deschaud, and F. Goulette, "Invariant EKF Design for Scan Matching-Aided Localization," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 6, pp. 2440–2448, Nov. 2015.
  - [36] S. Kim and J. Kim, "Occupancy Mapping and Surface Reconstruction Using Local Gaussian Processes With Kinect Sensors," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1335–1346, Oct. 2013.
  - [37] L. Maohai, S. Lining, H. Qingcheng, C. Zesu, and P. Songhao, "Robust Omnidirectional Vision based Mobile Robot Hierarchical Localization and Autonomous Navigation," *Inf. Technol. J.*, vol. 10, no. 1, pp. 29–39, Jan. 2011.
  - [38] L. Fernández, L. Payá, O. Reinoso, and L. M. Jimenez, "Appearance-based approach to hybrid metric-topological simultaneous localisation and mapping," *IET Intell. Transp. Syst.*, vol. 8, no. 8, pp. 688–699, 2014.
  - [39] S. R. Bista, P. R. Giordano, and F. Chaumette, "Appearance-Based Indoor Navigation by IBVS Using Line Segments," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 423–430, Jan. 2016.
  - [40] B. Bacca, X. Cufí, and J. Salví, "Vertical edge-based mapping using range-augmented omnidirectional vision sensor," *IET Comput. Vis.*, vol. 7, no. 2, pp. 135–143, Apr. 2013.
  - [41] E.-J. Jung, J. H. Lee, B.-J. Yi, J. Park, S. Yuta, and S.-T. Noh, "Development of a Laser-Range-Finder-Based Human Tracking and Control Algorithm for a Marathoner Service Robot," *IEEEASME Trans. Mechatron.*, vol. Early Access Online, 2014.
  - [42] Y. Liu and Y. Sun, "Mobile robot instant indoor map building and localization using 2D laser scanning data," in *2012 International Conference on System Science and Engineering (ICSSE)*, 2012, pp. 339–344.
  - [43] F. Bonaccorso, G. Muscato, and S. Baglio, "Laser range data scan-matching algorithm for mobile robot indoor self-localization," in *World Automation Congress (WAC), 2012*, 2012, pp. 1–5.
  - [44] Y. Nohara, T. Hasegawa, and K. Murakami, "Floor sensing system using laser range finder and mirror for localizing daily life commodities," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1030–1035.
  - [45] "http://eng.hagisonic.kr/cnt/prod/prod010102?uid=%2010&cateID=2."
  - [46] K.-W. Her, D.-H. Kim, and J.-E. Ha, "Localization of mobile robot using laser range finder and IR landmark," in *2012 12th International Conference on Control, Automation and Systems (ICCAS)*, 2012, pp. 459–461.
  - [47] S. Zhiwei, W. Yiyan, Z. Changjiu, and Z. Yi, "A new sensor fusion framework to deal with false detections for low-cost service robot localization," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 197–202.
  - [48] D.-H. Heo, A.-R. Oh, and T.-H. Park, "A localization system of mobile robots using artificial landmarks," in *2011 IEEE Conference on Automation Science and Engineering (CASE)*, 2011, pp. 139–144.
  - [49] M. L. Anjum *et al.*, "Sensor data fusion using Unscented Kalman Filter for accurate localization of mobile robots," in *2010 International Conference on Control Automation and Systems (ICCAS)*, 2010, pp. 947–952.



- 
- [50] E. Ivanjko and I. Petrovic, "Extended Kalman filter based mobile robot pose tracking using occupancy grid maps," in *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, 2004, vol. 1, pp. 311–314 Vol.1.
  - [51] S. Hong, T. Smith, F. Borrelli, and J. K. Hedrick, "Vehicle inertial parameter identification using Extended and unscented Kalman Filters," in *2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC)*, 2013, pp. 1436–1441.
  - [52] J. J. LaViola, "A comparison of unscented and extended Kalman filtering for estimating quaternion motion," in *American Control Conference, 2003. Proceedings of the 2003*, 2003, vol. 3, pp. 2435–2440 vol.3.
  - [53] G. Cotugno, L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese, "Extended and Unscented Kalman Filters for mobile robot localization and environment reconstruction," in *2013 21st Mediterranean Conference on Control Automation (MED)*, 2013, pp. 19–26.
  - [54] M. T. Nasri and W. Kinsner, "Extended and unscented Kalman filters for the identification of uncertainties in a process," in *2013 12th IEEE International Conference on Cognitive Informatics Cognitive Computing (ICCI\*CC)*, 2013, pp. 182–188.
  - [55] Y. Xu, S. L. Lau, R. Kusber, and K. David, "An experimental investigation of indoor localization by unsupervised Wi-Fi signal clustering," in *Future Network Mobile Summit (FutureNetw)*, 2012, 2012, pp. 1–10.
  - [56] M. D. František Duchoň, "Some Applications of Laser Rangefinder in Mobile Robotics," vol. 14, no. 2, 2012.
  - [57] S. Ikemoto, H. B. Amor, T. Minato, B. Jung, and H. Ishiguro, "Physical Human-Robot Interaction: Mutual Learning and Adaptation," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 24–35, Dec. 2012.
  - [58] C. A. Cifuentes, A. Frizera, R. Carelli, and T. Bastos, "Human-robot interaction based on wearable IMU sensor and laser range finder," *Robot. Auton. Syst.*, vol. 62, no. 10, pp. 1425–1439, Oct. 2014.
  - [59] S.-H. Tseng, Y. Chao, C. Lin, and L.-C. Fu, "Service robots: System design for tracking people through data fusion and initiating interaction with the human group by inferring social situations," *Robot. Auton. Syst.*, vol. 83, pp. 188–202, Sep. 2016.
  - [60] E. Hortal *et al.*, "SVM-based Brain-Machine Interface for controlling a robot arm through four mental tasks," *Neurocomputing*, vol. 151, Part 1, pp. 116–121, Mar. 2015.
  - [61] C. C. Tsai, Y. Z. Chen, and C. W. Liao, "Interactive emotion recognition using Support Vector Machine for human-robot interaction," in *IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009*, 2009, pp. 407–412.
  - [62] I.-J. Ding and J.-Y. Shi, "Kinect microphone array-based speech and speaker recognition for the exhibition control of humanoid robots," *Comput. Electr. Eng.*
  - [63] G. Canal, S. Escalera, and C. Angulo, "A Real-time Human-Robot Interaction system based on gestures for assistive scenarios," *Comput. Vis. Image Underst.*
  - [64] N. Yang *et al.*, "A study of the human-robot synchronous control system based on skeletal tracking technology," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 2191–2196.
  - [65] Y. Wang, G. Song, G. Qiao, Y. Zhang, J. Zhang, and W. Wang, "Wheeled robot control based on gesture recognition using the Kinect sensor," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 378–383.
  - [66] S. Boubou and E. Suzuki, "Classifying actions based on histogram of oriented velocity vectors," *J. Intell. Inf. Syst.*, vol. 44, no. 1, pp. 49–65, Jul. 2014.

- 
- [67]O. Patsadu, C. Nukoolkit, and B. Watanapa, "Human gesture recognition using Kinect camera," in *2012 International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2012, pp. 28–32.
  - [68]D. Xu, Y.-L. Chen, C. Lin, X. Kong, and X. Wu, "Real-time dynamic gesture recognition system based on depth perception for robot navigation," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 689–694.
  - [69]R. K. Megalingam, N. Saboo, N. Ajithkumar, S. Unny, and D. Menon, "Kinect based gesture controlled Robotic arm: A research work at HuT Labs," in *Innovation and Technology in Education (MITE), 2013 IEEE International Conference in MOOC*, 2013, pp. 294–299.
  - [70]V. H. Andaluz *et al.*, "Bilateral Virtual Control Human-Machine with Kinect Sensor," in *Andean Region International Conference (ANDESCON), 2012 VI*, 2012, pp. 101–104.
  - [71]P. M. Yanik *et al.*, "Use of kinect depth data and Growing Neural Gas for gesture based robot control," in *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, 2012, pp. 283–290.
  - [72]F. Mohammad, K. R. Sudini, V. Puligilla, and P. R. Kapula, "Tele-Operation of Robot Using Gestures," in *Modelling Symposium (AMS), 2013 7th Asia*, 2013, pp. 67–71.
  - [73]W. Song, X. Guo, F. Jiang, S. Yang, G. Jiang, and Y. Shi, "Teleoperation Humanoid Robot Control System Based on Kinect Sensor," in *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2012, vol. 2, pp. 264–267.
  - [74]G. Du and P. Zhang, "Markerless human–robot interface for dual robot manipulators using Kinect sensor," *Robot. Comput.-Integr. Manuf.*, vol. 30, no. 2, pp. 150–159, Apr. 2014.
  - [75]G. Chen and T. T. Pham, *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*, 1 edition. Boca Raton, FL: CRC Press, 2000.
  - [76]Y.-C. Lee, Z.-L. Lee, H.-H. Chiang, and T.-T. Lee, "Fuzzy-rule-based behavior control for collaborative human/robot navigation in unknown environments," in *2011 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2011, pp. 546–551.
  - [77]G. Liu, M. Yao, L. Zhang, and C. Zhang, "Fuzzy Controller for Obstacle Avoidance in Electric Wheelchair with Ultrasonic Sensors," in *2011 International Symposium on Computer Science and Society (ISCCS)*, 2011, pp. 71–74.
  - [78]S. Cui, X. Su, L. Zhao, Z. Bing, and G. Yang, "Study on ultrasonic obstacle avoidance of mobile robot based on fuzzy controller," in *2010 International Conference on Computer Application and System Modeling (ICCASM)*, 2010, vol. 4, pp. V4–233–V4–237.
  - [79]H. Saito, R. Amano, N. Moriyama, K. Kobayashi, and K. Watanabe, "Emergency obstacle avoidance module for mobile robots using a laser range finder," in *2013 Proceedings of SICE Annual Conference (SICE)*, 2013, pp. 348–353.
  - [80]V. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE Trans. Autom. Control*, vol. 31, no. 11, pp. 1058–1063, Nov. 1986.
  - [81]D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
  - [82]P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 188–195, Apr. 2005.

- 
- [83] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1192–1198.
  - [84] D. Hennes, D. Claes, K. Tuyls, and W. Meeussen, *Multi-robot collision avoidance with localization uncertainty*.
  - [85] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation. Proceedings*, 1985, vol. 2, pp. 500–505.
  - [86] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.
  - [87] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. Robot. Autom.*, vol. 20, no. 1, pp. 45–59, Feb. 2004.
  - [88] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: 'Follow the Gap Method,'" *Robot. Auton. Syst.*, vol. 60, no. 9, pp. 1123–1134, Sep. 2012.
  - [89] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots," in *1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings*, 1998, vol. 2, pp. 1572–1577 vol.2.
  - [90] I. Ulrich and J. Borenstein, "VFH\*: local obstacle avoidance with look-ahead verification," in *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, 2000, vol. 3, pp. 2505–2511 vol.3.
  - [91] J. Minguez, J. Osuna, and L. Montano, "A 'divide and conquer' strategy based on situations to achieve reactive collision avoidance in troublesome scenarios," in *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, 2004, vol. 4, pp. 3855–3862 Vol.4.
  - [92] J. W. Durham and F. Bullo, "Smooth Nearness-Diagram Navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, 2008, pp. 690–695.
  - [93] M. Mujahad, D. Fischer, B. Mertsching, and H. Jaddu, "Closest Gap based (CG) reactive obstacle avoidance Navigation for highly cluttered environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1805–1812.
  - [94] Y. Kim and S. Kwon, "A heuristic obstacle avoidance algorithm using vanishing point and obstacle angle," *Intell. Serv. Robot.*, pp. 1–9, Apr. 2015.
  - [95] Y. Hagiwara, T. Inamura, and Y. Choi, "Effectiveness evaluation of view-based navigation for obstacle avoidance," in *2013 13th International Conference on Control, Automation and Systems (ICCAS)*, 2013, pp. 1029–1033.
  - [96] J. Choi, D. Kim, H. Yoo, and K. Sohn, "Rear obstacle detection system based on depth from kinect," in *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2012, pp. 98–101.
  - [97] N. Abdelkrim, K. Issam, K. Lyes, and C. Khaoula, "Fuzzy logic controllers for Mobile robot navigation in unknown environment using Kinect sensor," in *2014 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2014, pp. 75–78.
  - [98] D. S. O. Correa, D. F. Sciotti, M. G. Prado, D. O. Sales, D. F. Wolf, and F. S. Osorio, "Mobile Robots Navigation in Indoor Environments Using Kinect Sensor," in *2012 Second Brazilian Conference on Critical Embedded Systems (CBSEC)*, 2012, pp. 36–41.
  - [99] H. Yue, W. Chen, X. Wu, and J. Zhang, "Kinect based real time obstacle detection for legged robots in complex environments," in *2013 8th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2013, pp. 205–210.

- 
- [100] M. Tanaka, "Robust parameter estimation of road condition by Kinect sensor," in *2012 Proceedings of SICE Annual Conference (SICE)*, 2012, pp. 197–202.
  - [101] A. Benzerrouk, L. Adouane, and P. Martinet, "Obstacle avoidance controller generating attainable set-points for the navigation of Multi-Robot System," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 487–492.
  - [102] B. Dhiyanesh, "Dynamic resource allocation for machine to cloud communications robotics cloud," in *2012 International Conference on Emerging Trends in Electrical Engineering and Energy Management (ICETEEEM)*, 2012, pp. 451–454.
  - [103] I. Stojmenovic, "Machine-to-Machine Communications With In-Network Data Aggregation, Processing, and Actuation for Large-Scale Cyber-Physical Systems," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 122–128, Apr. 2014.
  - [104] S. Hoshino, "Multi-robot coordination methodology in congested systems with bottlenecks," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2810–2816.
  - [105] D. E. Soltero, S. L. Smith, and D. Rus, "Collision avoidance for persistent monitoring in multi-robot systems with intersecting trajectories," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 3645–3652.
  - [106] J. Ng and T. Bräunl, "Performance Comparison of Bug Navigation Algorithms," *J. Intell. Robot. Syst.*, vol. 50, no. 1, pp. 73–84, Apr. 2007.
  - [107] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *1991 IEEE International Conference on Robotics and Automation, 1991. Proceedings*, 1991, pp. 1398–1404 vol.2.
  - [108] A. Ramakumar, U. Subramanian, and P. G. S. Prasanna, "High-throughput sample processing and sample management; the functional evolution of classical cytogenetic assay towards automation.," *Mutat. Res. Toxicol. Environ. Mutagen.*, vol. 793, pp. 132–141, Nov. 2015.
  - [109] E. S. Ruiz *et al.*, "Virtual and Remote Industrial Laboratory: Integration in Learning Management Systems," *IEEE Ind. Electron. Mag.*, vol. 8, no. 4, pp. 45–58, Dec. 2014.
  - [110] H. Liu, N. Stoll, S. Junginger, and K. Thurow, "A fast method for mobile robot transportation in life science automation," in *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*, 2013, pp. 238–242.
  - [111] M. Ghandour, H. Liu, N. Stoll, and K. Thurow, "Improving the navigation of indoor mobile robots using Kalman filter," in *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, 2015, pp. 1434–1439.
  - [112] M. Ghandour, H. Liu, N. Stoll, and K. Thurow, "Interactive collision avoidance system for indoor mobile robots based on human-robot interaction," in *2016 9th International Conference on Human System Interactions (HSI)*, 2016, pp. 209–215.
  - [113] M. Ghandour, H. Liu, N. Stoll, and K. Thurow, "A Hybrid Collision Avoidance System for Indoor Mobile Robots based on Human-Robot Interaction," presented at the The 17th International Conference on Mechatronics – Mechatronika 2016, Prague-Czech Republic, 2016.
  - [114] H. Liu, N. Stoll, S. Junginger, and K. Thurow, "A new approach to battery power tracking and predicting for mobile robot transportation using wavelet decomposition and ANFIS networks," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014, pp. 253–258.
  - [115] "Dr Robot Inc.: WiFi 802.11 robot, Network-based Robot, robotic, robot kit, humanoid robot, OEM solution." [Online]. Available: [http://www.drrobot.com/products\\_H20.asp](http://www.drrobot.com/products_H20.asp). [Accessed: 24-Nov-2016].

- 
- [116] Zhang, Y.D., Z. J. yang, H. M. Lu, X. X. Zhou, P. Phillips, Q. M. Liu, and S. H. Wang. "Facial Emotion Recognition Based on Biorthogonal Wavelet Entropy, Fuzzy Support Vector Machine, and Stratified Cross Validation." *IEEE Access* PP, no. 99 (2016): 1–1. doi:10.1109/ACCESS.2016.2628407.
  - [117] H. Schaathun, "Support Vector Machines," in *Machine Learning in Image Steganalysis*, Wiley-IEEE Press, 2012, pp. 179–196.
  - [118] V. Cherkassky and F. Mulier, "Support Vector Machines," in *Learning from Data: Concepts, Theory, and Methods*, Wiley-IEEE Press, 2007, pp. 404–466.
  - [119] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.
  - [120] J.-S. Gutmann and D. Fox, "An experimental comparison of localization methods continued," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002*, 2002, vol. 1, pp. 454–459 vol.1.
  - [121] A. Novel, B. R. Trilaksono, and R. A. Sasongko, "Guided Rocket Navigation design and implementation on Hardware in Loop Simulation," in *2013 3rd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME)*, 2013, pp. 249–254.
  - [122] J. Hu, H. Zhang, H. Huang, J. Feng, and G. Wang, "An improved Kalman filter algorithm base on quaternion correlation in object tracking," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2012, pp. 1725–1729.
  - [123] R. Saravanakumar and D. Jena, "Nonlinear estimation and control of wind turbine," in *2013 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2013, pp. 1–6.
  - [124] H. Liu, H. Tian, and Y. Li, "Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction," *Appl. Energy*, vol. 98, pp. 415–424, Oct. 2012.
  - [125] Y. Li, Z. Huang, N. Zhou, B. Lee, R. Diao, and P. Du, "Application of ensemble Kalman filter in power system state tracking and sensitivity analysis," in *Transmission and Distribution Conference and Exposition (T D), 2012 IEEE PES*, 2012, pp. 1–8.
  - [126] M. Mathe, S. P. Nandyala, and T. K. Kumar, "Speech enhancement using Kalman Filter for white, random and color noise," in *2012 International Conference on Devices, Circuits and Systems (ICDCS)*, 2012, pp. 195–198.
  - [127] J. D. Irwin, Ed., *The Industrial Electronics Handbook*, 1 edition. Berlin; Heidelberg: CRC Press, 1997.
  - [128] "Kinect for Windows SDK 2.0," *Microsoft Download Center*. [Online]. Available: <https://www.microsoft.com/en-us/download/details.aspx?id=44561>. [Accessed: 25-Feb-2016].
  - [129] T. Butkiewicz, "Low-cost coastal mapping using Kinect v2 time-of-flight cameras," in *Oceans - St. John's, 2014*, 2014, pp. 1–9.
  - [130] A. A. Abdulla, H. Liu, N. Stoll, and K. Thurow, "A New Robust Method for Mobile Robot Multifloor Navigation in Distributed Life Science Laboratories," *J. Control Sci. Eng.*, vol. 2016, 2016.
  - [131] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*, 2 edition. Amsterdam ; Boston, MA: Morgan Kaufmann, 2005.
  - [132] B. Coppin, *Artificial Intelligence Illuminated*, Computer ed. edition. Boston: Jones & Bartlett Learning, 2004.
  - [133] M. Negnevitsky, *Artificial Intelligence A Guide to Intelligent Systems*, Second. Addison Wesley.

- [134] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques, Third Edition*, 3 edition. Haryana, India; Burlington, MA: Morgan Kaufmann, 2011.
- [135] Z. Nematzadeh, R. Ibrahim, and A. Selamat, "Comparative studies on breast cancer classifications with k-fold cross validations using machine learning techniques," in *Control Conference (ASCC), 2015 10th Asian*, 2015, pp. 1–6.
- [136] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 569–575, Mar. 2010.
- [137] "Introduction to the C# Language and the .NET Framework." [Online]. Available: <https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx>. [Accessed: 06-Oct-2016].

## Appendix 1: The StarGazer Sensor

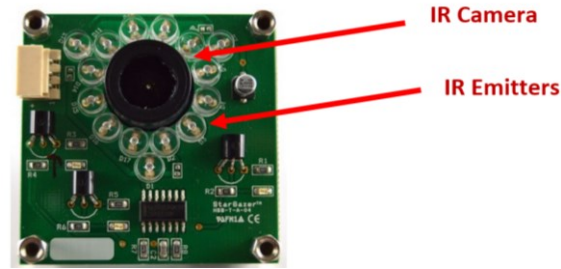


Figure A1.A: The IR sensor from Hagisonic- South Korea [45]

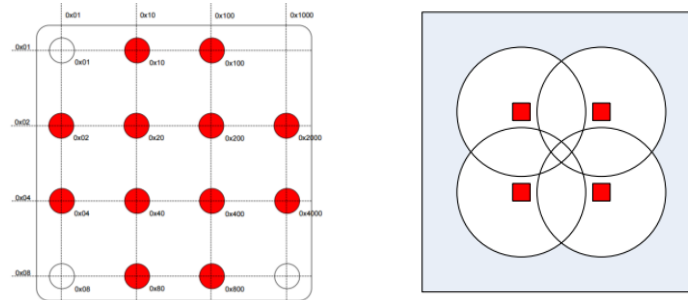


Figure A1.B: HL2 Landmark with the hexadecimal values and the best distribution [45]

Table A1.A: StarGazer Specifications [45]

Hardware Interface	UART(TTL 3.3V) 115,200bps
Size	50×50×28mm
Communication Protocol	User protocol based on ASCII code
Measurement Time	20 times/sec
Localization Range (per a Landmark)	2.5-5 m in diameter (for ceiling height 2-6m)
Repetitive Precision	2cm
Heading Angle Resolution	0.1 degree
Landmark Types (classification for ID numbers)	HL1: 31 ea (for a normal space) HL2: 4095 ea (for a larger space)

## Appendix 2

### The Kinect 2.0

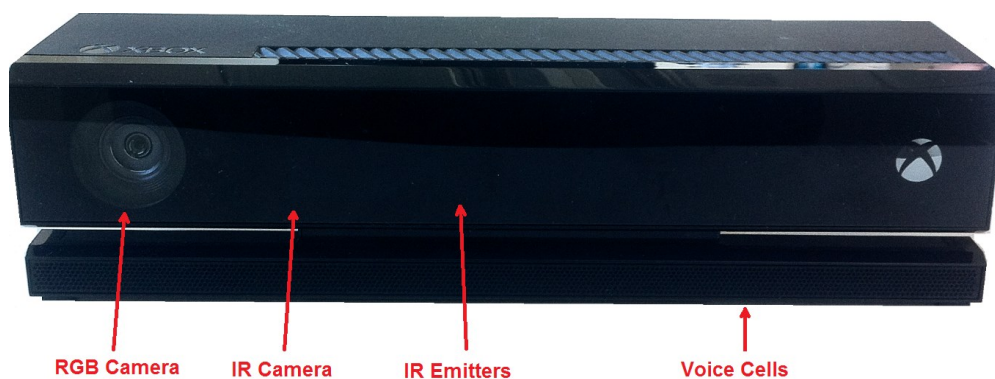


Figure A2.A: The Kinect V2 sensor

Table A2.A: Kinect 2.0 Specifications

Color Camera	1920x1080 – 30 fps
Depth Camea	512x424
Max Depth Distance	4.5 m
Min Depth Distance	50 cm
Horizontal Field of View	70°
Vertical Field of View	60°
Skeleton Joints Number	26
Number of Tracked Skeletons	6
Access Port	USB3



### Appendix 3

Table A3.A: The complete experiments for the SVM model

Experiment Number	C	Misclassified Samples for 8 folds	Train Time for 8 folds	SVM	model test time(ms)
1	4.53999E-05	397	112	21	<1
2	0.00012341	397	104	21	<1
3	0.000335463	397	103	21	<1
4	0.000911882	397	108	21	<1
5	0.002478752	397	107	21	<1
6	0.006737947	397	105	21	<1
7	0.018315639	355	102	21	<1
8	0.049787068	156	88	21	<1
9	0.135335283	33	71	21	<1
10	0.367879441	8	68	21	<1
11	1	9	56	21	<1
12	2.718281828	9	49	21	<1
13	7.389056099	8	45	21	<1
14	20.08553692	16	45	21	<1
15	54.59815003	16	44	21	<1
16	148.4131591	16	49	21	<1
17	403.4287935	16	42	21	<1
18	1096.633158	16	42	21	<1
19	2980.957987	16	45	21	<1
20	8103.083928	16	42	21	<1
21	22026.46579	16	44	21	<1
22	59874.14172	16	45	21	<1
23	162754.7914	16	44	21	<1
24	442413.392	16	44	21	<1
25	1202604.284	16	41	21	<1
26	3269017.372	16	41	21	<1
27	8886110.521	16	43	21	<1
28	24154952.75	16	44	21	<1
29	65659969.14	16	42	21	<1
30	178482301	16	42	21	<1
31	485165195.4	16	44	21	<1
32	1318815734	16	43	21	<1
33	3584912846	16	44	21	<1
34	9744803446	16	44	21	<1
35	26489122130	16	42	21	<1
36	72004899337	16	43	21	<1
37	1.9573E+11	16	42	21	<1
38	5.32048E+11	16	41	21	<1
39	1.44626E+12	16	44	21	<1
40	3.93133E+12	16	44	21	<1
41	1.06865E+13	16	42	21	<1
42	2.90488E+13	16	44	21	<1

43	7.8963E+13	16	44	21	<1
44	2.14644E+14	16	42	21	<1
45	5.83462E+14	16	42	21	<1
46	1.58601E+15	16	42	21	<1
47	4.31123E+15	16	47	21	<1
48	1.17191E+16	16	43	21	<1
49	3.18559E+16	16	43	21	<1
50	8.65934E+16	16	47	21	<1
51	2.35385E+17	16	43	21	<1
52	6.39843E+17	16	47	21	<1
53	1.73927E+18	16	44	21	<1
54	4.72784E+18	16	46	21	<1
55	1.28516E+19	16	41	21	<1
56	3.49343E+19	16	41	21	<1
57	9.49612E+19	16	45	21	<1
58	2.58131E+20	16	46	21	<1
59	7.01674E+20	16	49	21	<1
60	1.90735E+21	16	47	21	<1
61	5.18471E+21	16	46	21	<1
62	1.40935E+22	16	44	21	<1
63	3.83101E+22	16	44	21	<1
64	1.04138E+23	16	46	21	<1
65	2.83075E+23	16	46	21	<1
66	7.69479E+23	16	46	21	<1
67	2.09166E+24	16	45	21	<1
68	5.68572E+24	16	48	21	<1
69	1.54554E+25	16	44	21	<1
70	4.20121E+25	16	53	21	<1
71	1.14201E+26	16	47	21	<1
72	3.1043E+26	16	43	21	<1
73	8.43836E+26	16	48	21	<1
74	2.29378E+27	16	43	21	<1
75	6.23515E+27	16	45	21	<1
76	1.69489E+28	16	41	21	<1
77	4.60719E+28	16	44	21	<1
78	1.25236E+29	16	44	21	<1
79	3.40428E+29	16	42	21	<1
80	9.25378E+29	16	44	21	<1

## Appendix4

### The experiments for the human-robot interaction system

Table A4.A: The experiments for the first person (height 165)

Depth (Z)	Distance (X)	Gesture	Angle	Time	Note
1.8	-0.6	Right	-40	377	
	0	Master	-20	365	
	0.6	Left	0	392	Forward wrong “Kinect failure”
2.4	-1.15	Resume	20	359	
	-0.57	Forward	40	396	
	0	Stop	-40	372	
	0.57	backward	-20	361	
	1.15	Forward	0	375	
3.2	-1.64	Left	20	346	
	-0.8	backward	40	773	
	0	Master	-40	369	
	0.8	Resume	-20	367	
	1.64	Stop	0	371	
4.0	-2.2	Right	20	377	
	-1.1	Stop	40	383	
	0	Resume	-40	348	
	1.1	Master	-20	363	
	2.2	Left	0	381	

Table A4.B: The experiments for the second person (height 170)

Depth (Z)	Distance (X)	Gesture	Angle	Time	Note
1.8	-0.6	Resume	-20	380	
	0	Forward	0	773	State transition
	0.6	Stop	20	382	
2.4	-1.15	backward	40	766	State transition
	-0.57	Forward	-40	359	
	0	Left	-20	368	
	0.57	Right	0	378	
	1.15	Master	20	358	
3.2	-1.64	Resume	40	380	
	-0.8	Stop	-40	367	
	0	Right	-20	355	
	0.8	Left	0	368	
	1.64	Resume	20	368	
4.0	-2.2	Backward	40	368	
	-1.1	Master	-40	365	Left wrong "Kinect failure"
	0	Right	-20	752	State transition
	1.1	Stop	0	376	
	2.2	Forward	-20	349	

Table A4.C: The experiments for the third person (height 177)

Depth (Z)	Distance (X)	Gesture	Angle	Time	Note
1.8	-0.6	Master	0	374	
	0	Stop	20	361	
	0.6	Backward	40	383	
2.4	-1.15	Resume	-40	349	
	-0.57	Left	-20	354	
	0	Forward	0	360	
	0.57	Left	20	359	
	1.15	Backward	40	353	
3.2	-1.64	Master	-40	354	
	-0.8	Stop	-20	355	
	0	Right	0	346	
	0.8	Left	20	339	
	1.64	Resume	40	346	
4	-2.2	Master	-40	375	
	-1.1	Resume	-20	347	
	0	Right	0	349	
	1.1	Backward	-20	735	State transition
	2.2	Forward	-40	348	

Table A4.D: The experiments for the fourth person (height 179)

Depth (Z)	Distance (X)	Gesture	Angle	Time	Note
1.8	-0.6	Stop	20	379	
	0	Right	40	354	
	0.6	Left	-40	381	
2.4	-1.15	Backward	-20	337	
	-0.57	Master	0	358	
	0	Resume	20	362	
	0.57	Forward	40	379	
	1.15	Right	-40	350	
3.2	-1.64	Stop	-20	744	State transition
	-0.8	Left	0	386	
	0	Backward	20	397	
	0.8	Right	40	356	
	1.64	Stop	-40	344	
4	-2.2	Resume	-20	351	
	-1.1	Backward	0	794	
	0	Left	-20	347	
	1.1	Master	-40	368	
	2.2	Forward	20	381	

Table A4.E: The experiments for the fifth person (height 188)

Depth (Z)	Distance (X)	Gesture	Angle	Time	Note
1.8	-0.6	Master	0	369	
	0	Stop	20	384	
	0.6	Forward	40	347	
2.4	-1.15	Backward	-40	358	
	-0.57	Right	-20	374	
	0	Left	0	366	
	0.57	Resume	20	752	State Transition
	1.15	Master	40	382	
3.2	-1.64	Stop	-40	377	
	-0.8	Forward	-20	354	
	0	Backward	0	373	
	0.8	Right	20	376	
	1.64	Left	40	362	
4	-2.2	Resume	-20	372	
	-1.1	Master	-40	357	
	0	Stop	0	363	
	1.1	Forward	20	359	
	2.2	Backward	40	372	

## Appendix5

### The programming code for “Autonomous Collision Avoidance” Function

```

public void AutonomousCollisionAvoidance()
{
    Dynamic_CA_InProgress = true;
    Tag_AutonomousInProcess = true;
    CA_TimeCycle_Finished = false;
    Timer_DCA_Time_Finished = false;
    Timer_DCA.Stop();
    Timer_CA.Stop();
    bool ThereisRegion = false;
    double goal_angle;
    CA_TimeCycle_Finished = false;
    LogFileData += "\r\n" + "ACA";

    goal_angle = (Math.Atan((X_WP - X_Current) / (Y_WP - Y_Current))) * 180 / 3.14;

    if (ThereIsSomeone && Array_X_DCA[0] != 0)
    {
        ClientS = "H20-RBC-V2";
        CommandNameS = "CAParameters";

        int c = 0;
        for (int i = 0; i < 6; i++)
        {
            if (Array_X_DCA[i] != 0)
                c++;
        }

        double[] X_People_DCA_Arranged = new double[c];
        double[] Z_People_DCA_Arranged = new double[c];
        double[] Radius_People_DCA_Arranged = new double[c];
        for (int i = 0; i < c; i++)
        {
            X_People_DCA_Arranged[i] = Array_X_DCA[i];
            Z_People_DCA_Arranged[i] = Array_Z_DCA[i];
            Radius_People_DCA_Arranged[i] = Array_Human_Radius_DCA[i];
        }
        bool didSwap;
        do
        {
            didSwap = false;
            if (X_People_DCA_Arranged.Length > 1)
            {
                for (int i = 0; i < X_People_DCA_Arranged.Length - 1; i++)
                {
                    if (X_People_DCA_Arranged[i] > X_People_DCA_Arranged[i + 1])
                    {
                        double temp = X_People_DCA_Arranged[i + 1];
                        double temp1 = Z_People_DCA_Arranged[i + 1];
                        double temp2 = Radius_People_DCA_Arranged[i + 1];
                        X_People_DCA_Arranged[i + 1] = X_People_DCA_Arranged[i];
                        Z_People_DCA_Arranged[i + 1] = Z_People_DCA_Arranged[i];
                        Radius_People_DCA_Arranged[i + 1] = Radius_People_DCA_Arranged[i];
                        X_People_DCA_Arranged[i] = temp;
                        Z_People_DCA_Arranged[i] = temp1;
                        Radius_People_DCA_Arranged[i] = temp2;
                        didSwap = true;
                    }
                }
            }
        } while (didSwap);

        for (int i = 0; i < X_People_DCA_Arranged.Length; i++)
    }

```

```

    {
        X_People_DCA_Arranged[i] = X_People_DCA_Arranged[i] + X_Correction_factor;
    }
    double[] People_Angles_Arranged = new double[c];

    for (int i = 0; i < X_People_DCA_Arranged.Length; i++)
    {
        People_Angles_Arranged[i] = Math.Atan(X_People_DCA_Arranged[i] /
        Z_People_DCA_Arranged[i]);
    }
    double[] Gaps_X;
    double[] Gaps_Z;
    double[] Gaps_Radius;
    double[] Gaps_AngularWidth;
    double[] Gaps_TheFarX;
    double[] Gaps_TheAngleofMiddle;
    double[] Direction;
    int NumberOfGaps = 0;

    if (X_People_DCA_Arranged.Length > 1)
    {
        for (int i = 0; i < X_People_DCA_Arranged.Length - 1; i++)
        {
            if ((X_People_DCA_Arranged[i + 1] - X_People_DCA_Arranged[i]) > ((2 * Rr +
            Radius_People_DCA_Arranged[i + 1] + Radius_People_DCA_Arranged[i]) +
            Safety_Displacement))
            {
                NumberOfGaps++;
            }
        }
    }

    double Gaps_XLeft, Gaps_XRight, Gaps_XAllowedLeft, Gaps_XAllowedRight;
    Gaps_XLeft = X_People_DCA_Arranged[0] - Radius_People_DCA_Arranged[0] - Rr -
    Safety_Displacement / 2;
    Gaps_XRight = X_People_DCA_Arranged[X_People_DCA_Arranged.Length - 1] +
    Radius_People_DCA_Arranged[Radius_People_DCA_Arranged.Length - 1] +
    Rr + Safety_Displacement / 2;
    Gaps_XAllowedLeft = -0.7 * Z_People_DCA_Arranged[0];
    Gaps_XAllowedRight = 0.7 * Z_People_DCA_Arranged[Z_People_DCA_Arranged.Length - 1];
    if (Gaps_XAllowedLeft < Gaps_XLeft)
    {
        NumberOfGaps = NumberOfGaps + 1;
    }
    if (Gaps_XAllowedRight > Gaps_XRight)
    {
        NumberOfGaps = NumberOfGaps + 1;
    }

    Gaps_X = new double[NumberOfGaps];
    Gaps_Z = new double[NumberOfGaps];
    Gaps_Radius = new double[NumberOfGaps];
    Gaps_AngularWidth = new double[NumberOfGaps];
    Gaps_TheFarX = new double[NumberOfGaps];
    Gaps_TheAngleofMiddle = new double[NumberOfGaps];
    Direction = new double[NumberOfGaps];
    if (NumberOfGaps > 0)
    {
        int counter = 0;
        ThereisRegion = true;
        if (Gaps_XAllowedLeft < Gaps_XLeft)
        {
            Gaps_X[0] = X_People_DCA_Arranged[0] - Radius_People_DCA_Arranged[0]
            - Rr - Safety_Displacement / 2;
            Gaps_Z[0] = Z_People_DCA_Arranged[0];
            Gaps_Radius[0] = Radius_People_DCA_Arranged[0];
            Gaps_AngularWidth[0] = Math.Abs(-0.61 - People_Angles_Arranged[0]);
            Gaps_TheFarX[0] = X_People_DCA_Arranged[0];

```

```

        Gaps_TheAngleofMiddle[0] = Math.Atan(Gaps_X[0] / Gaps_Z[0]) + (-0.61
            - Math.Atan(Gaps_X[0] / Gaps_Z[0])) / 2;
        Direction[0] = Math.Atan(Gaps_X[0] / Gaps_Z[0]);
        counter++;
    }
    for (int i = 0; i < X_People_DCA_Arranged.Length - 1; i++)
    {
        if ((X_People_DCA_Arranged[i + 1] - X_People_DCA_Arranged[i]) > (2 *
            Rr + Radius_People_DCA_Arranged[i] + Radius_People_DCA_Arranged[i +
            1] + Safety_Displacement))
        {
            Gaps_X[counter] = (X_People_DCA_Arranged[i] +
                X_People_DCA_Arranged[i + 1]) / 2;
            Gaps_AngularWidth[counter] = People_Angles_Arranged[i + 1] -
                People_Angles_Arranged[i];
            if (Z_People_DCA_Arranged[i] > Z_People_DCA_Arranged[i + 1])
            {
                Gaps_Z[counter] = Z_People_DCA_Arranged[i];
                Gaps_Radius[counter] = Radius_People_DCA_Arranged[i];
                Gaps_TheFarX[counter] = X_People_DCA_Arranged[i];
            }
            else
            {
                Gaps_Z[counter] = Z_People_DCA_Arranged[i + 1];
                Gaps_Radius[counter] = Radius_People_DCA_Arranged[i + 1];
                Gaps_TheFarX[counter] = X_People_DCA_Arranged[i + 1];
            }
            Gaps_TheAngleofMiddle[counter] = Math.Atan(Gaps_X[counter] /
                Gaps_Z[counter]);
            Direction[counter] = Math.Atan(Gaps_X[counter] /
                Gaps_Z[counter]);
            counter++;
        }
    }
    if (Gaps_XAloowedRight > Gaps_XRight)
    {
        Gaps_X[NumberOfGaps - 1] =
            X_People_DCA_Arranged[X_People_DCA_Arranged.Length - 1] +
            Radius_People_DCA_Arranged[X_People_DCA_Arranged.Length - 1] + Rr +
            Safety_Displacement / 2;

        Gaps_Z[NumberOfGaps - 1] =
            Z_People_DCA_Arranged[Z_People_DCA_Arranged.Length - 1];

        Gaps_Radius[NumberOfGaps - 1] =
            Radius_People_DCA_Arranged[Radius_People_DCA_Arranged.Length - 1];
        Gaps_AngularWidth[NumberOfGaps - 1] = Math.Abs(0.61 -
            People_Angles_Arranged[People_Angles_Arranged.Length - 1]);
        Gaps_TheFarX[NumberOfGaps - 1] =
            X_People_DCA_Arranged[X_People_DCA_Arranged.Length - 1];
        Gaps_TheAngleofMiddle[NumberOfGaps - 1] =
            Math.Atan(Gaps_X[NumberOfGaps - 1] / Gaps_Z[NumberOfGaps - 1]) +
            (0.61 - Math.Atan(Gaps_X[NumberOfGaps - 1] / Gaps_Z[NumberOfGaps -
            1])) / 2;
        Direction[NumberOfGaps - 1] = Math.Atan(Gaps_X[NumberOfGaps - 1] /
            Gaps_Z[NumberOfGaps - 1]);
    }
}
else
    ThereisRegion = false;

if (ThereisRegion)
{
    double XToAvoid = 0;
    double ZToAvoid = 0;
    double RadiusToAvoid = 0;
    double RegionWidth_Angle = 0;

```



```

double X_FarPerson = 0;
for (int i = 0; i < Gaps_TheAngleofMiddle.Length; i++)
{
    Gaps_TheAngleofMiddle[i] = Gaps_TheAngleofMiddle[i] * 180 / 3.14;
}
int pointer_to_selected_angle = 0;
double temp_gap_anglevalue = 100;
for (int i = 0; i < Gaps_TheAngleofMiddle.Length; i++)
{
    if (Gaps_TheAngleofMiddle[i] > goal_angle)
    {
        double tem = Gaps_TheAngleofMiddle[i] - goal_angle;
        if (tem < temp_gap_anglevalue)
        {
            pointer_to_selected_angle = i;
            temp_gap_anglevalue = tem;
        }
    }
    else
    {
        double tem = goal_angle - Gaps_TheAngleofMiddle[i];
        if (tem < temp_gap_anglevalue)
        {
            pointer_to_selected_angle = i;
            temp_gap_anglevalue = tem;
        }
    }
}

XToAvoid = Gaps_X[pointer_to_selected_angle];
ZToAvoid = Gaps_Z[pointer_to_selected_angle];
RadiusToAvoid = Gaps_Radius[pointer_to_selected_angle];
RegionWidth_Angle = Gaps_AngularWidth[pointer_to_selected_angle];
X_FarPerson = Gaps_TheFarX[pointer_to_selected_angle];

double Z_ToAvoidNEWNEW = Math.Sqrt(XToAvoid * XToAvoid + ZToAvoid * ZToAvoid);
double theta = Math.Atan(XToAvoid / ZToAvoid) * 180 / 3.14;
double d = Z_ToAvoidNEWNEW + 2 * Rr + RadiusToAvoid;
double Linear_Velocity_Factor1 = 0;
double Angular_Velocity_Factor1 = 0;
double maximum_linear_velocity = (1000 / Linear_Velocity_Factor_DCA);
double velocity = maximum_linear_velocity * (Math.Log10
    (RegionWidth_Angle * (180 / 3.14))) / Math.Log10(70));
Linear_Velocity_Factor1 = Math.Round(1000 / velocity);
double maximum_angular_velocity = (1000 / Angular_Velocity_Factor_DCA);
double angular_velocity = maximum_angular_velocity * (Math.Log10(Math.Abs
    (RegionWidth_Angle * (180 / 3.14))) / Math.Log10(70));
Angular_Velocity_Factor1 = Math.Round(1000 / angular_velocity);
DepthS = Convert.ToString(0);
string d11 = d.ToString("F2");
DistanceS = Convert.ToString(0);
AngleS = Convert.ToString(Math.Round(theta));
ID_Counter++;
IDS = Convert.ToString(ID_Counter);
TimeSDistance = Convert.ToString(0);
TimeSRotation = Convert.ToString(Math.Round(Math.Abs(theta *
    Angular_Velocity_Factor1)));

XMLDataToSend se = new XMLDataToSend(this);
byte[] ss = se.XMLPrepare();
HRI_Server.SendOutData(ss);
Thread.Sleep(Convert.ToInt32(Math.Abs(theta * Angular_Velocity_Factor1)));
DistanceS = AngleS = DepthS = TimeSDistance = TimeSRotation = IDS = "";
Array.Clear(ss, 0, ss.Length);
DepthS = Convert.ToString(Math.Round(ZToAvoid));
DistanceS = Convert.ToString(Convert.ToDouble(d11));
AngleS = Convert.ToString(0);

```

---

```

ID_Counter++;
IDS = Convert.ToString(ID_Counter);
TimeSDistance = Convert.ToString(Math.Round(Math.Abs(d *
    Linear_Velocity_Factor1)));
TimeSRotation = Convert.ToString(0);
se = new XMLDataToSend(this);
ss = se.XMLPrepare();
HRI_Server.SendOutData(ss);
Thread.Sleep(Convert.ToInt32((Math.Abs(d * Linear_Velocity_Factor1))));
DistanceS = AngleS = DepthS = TimeSDistance = TimeSRotation = IDS = "";
Array.Clear(ss, 0, ss.Length);
}
else
{
    ThereisRegion = false;
    synth.SpeakAsync("No Free path");
    SetText_Plus(TheRaisedFunction, "ACA -- No Free Path" + "\r\n");
}
Dynamic_CA_InProgress = false;
CA_TimeCycle_Finshed = true;
Tag_AutonomousInProcess = false;
Timer_CA.Start();
}
else
{
    Dynamic_CA_InProgress = false;
    CA_TimeCycle_Finshed = true;
    Tag_AutonomousInProcess = false;
}
}

```

## Appendix6

### The experiments for the collision avoidance system

Table A6.A: Experimental results for “move forward”

Experiment no	Distance(m)	Velocity(m/s)	Time (ms)	Note
1	1.2	0.1	12000	
2	1.2	0.1	12000	
3	1.2	0.1	12000	
4	1.2	0.1	12000	
5	1.2	0.1	12000	
6	1.2	0.1	12000	
7	1.2	0.1	12000	
8	1.2	0.1	12000	
9	1.2	0.1	12000	
10	1.2	0.1	12000	
11	1.2	0.2	6000	
12	1.2	0.2	6000	
13	1.2	0.2	6000	
14	1.2	0.2	6000	
15	1.2	0.2	6000	
16	1.2	0.2	6000	
17	1.2	0.2	6000	
18	1.2	0.2	6000	
19	1.2	0.2	6000	
20	1.2	0.2	6000	
21	1.2	0.3	4000	Deviation 4°
22	1.2	0.3	4000	Deviation 4°
23	1.2	0.3	4000	Deviation 3°
24	1.2	0.3	4000	Deviation 3°
25	1.2	0.3	4000	Deviation 2°
26	1.2	0.3	4000	Deviation 4°
27	1.2	0.3	4000	Deviation 3°
28	1.2	0.3	4000	Deviation 2°
29	1.2	0.3	4000	Deviation 4°
30	1.2	0.3	4000	Deviation 3°

Table A6.B: Experimental results for “Move Backward”

Experiment no	Distance(m)	Velocity(m/s)	Time (ms)	Note
1	1.2	0.1	12000	
2	1.2	0.1	12000	
3	1.2	0.1	12000	
4	1.2	0.1	12000	
5	1.2	0.1	12000	
6	1.2	0.1	12000	
7	1.2	0.1	12000	
8	1.2	0.1	12000	
9	1.2	0.1	12000	
10	1.2	0.1	12000	
11	1.2	0.2	6000	
12	1.2	0.2	6000	
13	1.2	0.2	6000	
14	1.2	0.2	6000	
15	1.2	0.2	6000	
16	1.2	0.2	6000	
17	1.2	0.2	6000	
18	1.2	0.2	6000	
19	1.2	0.2	6000	
20	1.2	0.2	6000	
21	1.2	0.3	4000	Deviation 3°
22	1.2	0.3	4000	Deviation 4°
23	1.2	0.3	4000	Deviation 2°
24	1.2	0.3	4000	Deviation 2°
25	1.2	0.3	4000	Deviation 3°
26	1.2	0.3	4000	Deviation 4°
27	1.2	0.3	4000	Deviation 4°
28	1.2	0.3	4000	Deviation 3°
29	1.2	0.3	4000	Deviation 4°
30	1.2	0.3	4000	Deviation 2°

Table A6.C: Experimental results for “move right”  $v_{max} = 0.2 \text{ m/s}$   $\omega_{max} = 0.4 \text{ rad/s}$ 

No.	Region Candidate	Region Candidate	Heading Angle	d (m)	$\theta_r$ degree	V m/s	$\omega$ rad/s	Position	Time (ms)	Note
1	-3.7	25.4	-3.7	3.03	34.9	0.167	0.335	Middle	18316	
2	9.5	-----	9.5	2.96	33.0	0.164	0.329	Middle	18530	
3	8.7	-----	8.7	2.92	42.6	0.176	0.356	Middle	16980	
4	2.7	30.2	2.7	2.94	33.4	0.165	0.329	Middle	17997	
5	24.6	-----	24.6	3.09	24.4	0.150	0.300	Middle	21954	
6	22.7	-----	22.7	3.13	26.1	0.153	0.306	Right	21676	
7	33.4	-----	33.4	3.31	13.6	0.122	0.245	Right	29326	
8	20.0	-----	20.0	2.92	30.6	0.161	0.323	Right	19243	Missed skeleton
9	22.9	-----	22.9	3.65	23.3	0.148	0.295	Right	25995	
10	16.0	-----	16.0	3.37	31.4	0.162	0.323	Right	21633	

Table A6.D: Experimental results for “move right”  $v_{max} = 0.25 \text{ m/s}$   $\omega_{max} = 0.3 \text{ rad/s}$ 

No.	Region Candidate	Region Candidate	Heading Angle	d (m)	$\theta_r$ degree	V m/s	$\omega$ rad/s	Position	Time (ms)	Note
1	-6.4	25.0	-6.4	2.92	34.9	0.209	0.249	Middle	14443	
2	1.16	31.1	1.16	2.95	38.7	0.215	0.256	Middle	13822	
3	7.1	-----	7.1	3.16	42.4	0.220	0.264	Middle	14796	
4	13.9	-----	13.9	3.03	40.4	0.217	0.260	Middle	14873	
5	-0.98	24.7	-0.98	3.50	36.5	0.211	0.252	Middle	16596	
6	12.7	-----	12.7	3.17	36.2	0.211	0.252	Right	15890	
7	15.4	-----	15.4	3.50	30.8	0.201	0.242	Right	18497	
8	26.9	-----	26.9	3.44	19.9	0.176	0.210	Right	21789	
9	11.1	-----	11.1	3.20	35.8	0.210	0.252	Right	16007	Missed skeleton
10	26.9	-----	26.9	3.50	20.2	0.176	0.212	Right	22043	

Table A6.E: Experimental results for “move right”  $v_{max} = 0.3 \text{ m/s}$   $\omega_{max} = 0.5 \text{ rad/s}$ 

No.	Region Candidate	Region Candidate	Heading Angle	d (m)	$\theta_r$ degree	V m/s	$\omega$ rad/s	Position	Time (ms)	Note
1	-5.4	24.8	-5.4	3.02	37.0	0.255	0.425	Middle	12071	
2	9.0	-----	9.0	3.34	37.1	0.255	0.425	Middle	13473	
3	15.6	-----	15.6	3.35	36.8	0.254	0.425	Middle	13830	Dev 3°
4	-0.4	30.8	-0.4	3.12	41.1	0.262	0.436	Middle	11903	
5	20.5	-----	20.5	3.54	26.3	0.231	0.387	Middle	16248	Dev 4°
6	11.9	-----	11.9	3.18	36.6	0.254	0.425	Right	13000	
7	20.7	-----	20.7	3.22	27.8	0.234	0.387	Right	14682	Missed skeleton
8	24.1	-----	24.1	3.14	24.6	0.226	0.379	Right	14978	Dev 3°
9	25.2	-----	25.2	3.35	22.9	0.221	0.371	Right	16358	
10	14.5	-----	14.5	3.83	31.5	0.243	0.405	Right	16337	

Table A6.F: Experimental results for “move left”  $v_{max} = 0.15 \text{ m/s}$   $\omega_{max} = 0.3 \text{ rad/s}$ 

No.	Region Candidate	Region Candidate	Heading Angle	d (m)	$\theta_r$ degree	V m/s	$\omega$ rad/s	Position	Time (ms)	Note
1	-29.5	-5.9	-5.9	3.10	33.8	0.124	0.249	Middle	25420	
2	-13.0	-----	-13.0	2.69	47.0	0.135	0.272	Middle	20642	
3	-16.8	8.2	8.2	3.57	30.2	0.120	0.242	Middle	30256	
4	-11.9	-----	-11.9	3.19	33.8	0.124	0.249	Middle	26523	
5	-28.1	-0.2	-0.2	2.99	32.2	0.122	0.245	Middle	24455	
6	-12.4	-----	-12.4	2.65	39.9	0.130	0.260	Right	21200	Missed skeleton
7	-13.5	-----	-13.5	2.61	40.1	0.130	0.260	Right	20996	
8	-25.9	-----	-25.9	3.49	20.5	0.106	0.212	Right	34886	
9	-6.2	-----	-6.2	3.10	41.8	0.131	0.264	Right	23961	
10	-13.9	-----	-13.9	2.97	36.4	0.126	0.252	Right	24414	

Table A6.G: Experimental results for “move left”  $v_{max} = 0.2 \text{ m/s}$   $\omega_{max} = 0.4 \text{ rad/s}$ 

No.	Region Candidate	Region Candidate	Heading Angle	d (m)	$\theta_r$ degree	V m/s	$\omega$ rad/s	Position	Time (ms)	Note
1	-19.4	4.3	4.3	3.33	33.5	0.165	0.329	Middle	20412	Missed skeleton
2	-26.9	2.6	2.6	2.78	36.9	0.169	0.342	Middle	16525	
3	-23.4	7.8	7.8	2.77	36.2	0.168	0.335	Middle	16863	
4	-30.6	0.4	0.4	2.64	40.0	0.173	0.348	Middle	15241	
5	-21.6	8.5	8.5	2.74	33.9	0.165	0.329	Middle	17013	
6	-17.0	-----	-17.0	3.23	30.8	0.161	0.323	Right	20951	
7	-7.5	-----	-7.5	2.88	41.8	0.175	0.348	Right	16789	
8	-11.7	-----	-11.7	3.16	35.7	0.168	0.335	Right	19410	
9	-24.6	-----	-24.6	3.38	21.1	0.143	0.285	Right	25067	
10	-25.9	-----	-25.9	3.08	22.0	0.145	0.290	Right	22706	

Table A6.H: Experimental results for “move left”  $v_{max} = 0.3 \text{ m/s}$   $\omega_{max} = 0.25 \text{ rad/s}$ 

No.	Region Candidate	Region Candidate	Heading Angle	d (m)	$\theta_r$ degree	V m/s	$\omega$ rad/s	Position	Time (ms)	Note
1	-30.2	-1.9	-1.9	2.82	37.2	0.255	0.212	Middle	11221	
2	-25.2	3.6	3.6	2.80	35.2	0.251	0.210	Middle	11472	
3	-12.9	-----	-12.9	2.78	36.5	0.254	0.212	Middle	12030	Dev 4°
4	-34.3	-8.9	-8.9	2.95	36.4	0.253	0.212	Middle	12368	Missed skeleton
5	-6.7	-----	-6.7	2.79	39.1	0.259	0.215	Middle	11351	
6	-9.8	-----	-9.8	3.17	38.3	0.257	0.215	Right	13128	
7	-6.6	-----	-6.6	2.98	41.2	0.262	0.218	Right	11890	Dev 3°
8	-10.8	-----	-10.8	2.98	38.3	0.257	0.215	Right	12485	
9	-26.3	-----	-26.3	2.94	21.6	0.217	0.181	Right	16102	
10	-12.1	-----	-12.12	2.90	36.8	0.254	0.212	Right	12395	

Table A6.I: Experimental results for Autonomous CA  $v_{max} = 0.15 \text{ m/s}$   $\omega_{max} = 0.2 \text{ rad/s}$

No	Region Cand	Region Cand	Region Cand	Waypoint	Head Angle	d (m)	$\theta_r$	V	$\omega$	Time	Note
1	-29.9	5.2	-----	12.7	5.2	2.66	37.4	0.127	0.171	21339	
2	-18.7	12.45	-----	-86.9	-18.7	2.9	31.5	0.121	0.163	25976	
3	-33.4	23.7	-----	-75.0	-33.4	2.84	16.5	0.099	0.13	33094	
4	-18.03	5.12	-----	80.03	5.12	2.85	41.69	0.131	0.176	22178	
5	-31.0	22.1	-----	0.88	22.1	3.12	26.7	0.11	0.15	29420	
6	-21.7	11.13	-----	-59.6	-21.7	2.79	29.2	0.12	0.158	25888	
7	-19.6	17.6	-----	80.8	17.6	2.51	36.0	0.126	0.169	21647	
8	-16.77	13.32	-----	8.82	13.32	2.93	36.8	0.127	0.169	24448	
9	-16.7	17.2	-----	8.75	17.2	2.69	34.7	0.125	0.167	23343	Missed skeleton
10	-33.2	-3.9	-----	-35.1	-33.2	2.92	15	0.09	0.127	35105	

Table A6.J: Experimental results for Autonomous CA  $v_{max} = 0.2 \text{ m/s}$   $\omega_{max} = 0.5 \text{ rad/s}$

No	Region Cand	Region Cand	Region Cand	Waypoint	Head Angle	d (m)	$\theta_r$ deg	V m/s	$\omega$ rad/s	Time (ms)	Note
1	-30.5	3.2	34.5	8.3	3.2	2.69	37.9	0.171	0.425	15866	
2	-32.1	23.2	-----	35.8	23.2	3.17	25.5	0.152	0.379	21895	
3	-29.9	22.5	-----	17.4	22.5	3.04	27.3	0.155	0.387	20551	
4	-22.1	9.4	34.6	13.9	9.4	3.16	31.6	0.162	0.405	19881	
5	-32.7	19.5	-----	-78.8	-32.7	3.12	15.9	0.130	0.323	25796	Missed skeleton
6	-32.9	22.1	-----	-87.4	-32.9	2.86	16.8	0.132	0.329	23309	
7	-18.0	16.3	-----	-84.1	-18.0	2.75	34.0	0.166	0.415	17359	
8	-18.7	23.0	-----	-8.1	-18.7	2.34	37.3	0.170	0.425	14552	
9	-25.3	9.2	33.6	84.4	33.6	3.54	12.3	0.118	0.295	32023	
10	-28.5	-----	-----	-81.8	-28.5	2.49	25.7	0.152	0.379	17623	



Table A6.K: Experimental results for Autonomous  $v_{max} = 0.3 \text{ m/s}$   $\omega_{max} = 0.25 \text{ rad/s}$

No	Region Cand	Region Cand	Region Cand	Wayp oint	Head Angle	d (m)	$\theta_r$ deg	V m/s	$\omega$ rad/s	Time (ms)	Note
1	-30.6	3.3	34.7	71.6	34.7	3.25	12.3	0.177	0.147	22454	Missed skeleton
2	21.7	-----	-----	-86.6	21.7	2.61	31.3	0.243	0.202	12619	
3	-12.4	19.5	-----	-89.2	-12.4	2.85	38.8	0.258	0.215	12064	
4	-13.9	18.6	-----	3.7	18.6	2.88	32.4	0.245	0.205	13339	
5	-26.8	32.8	-----	-69.6	-26.8	2.81	23.4	0.222	0.185	15152	Dev 4°
6	-25.8	5.6	33.8	2.4	5.6	2.93	34.4	0.25	0.207	12203	
7	-31.3	25.5	-----	-84.5	-31.3	3.09	16.6	0.198	0.166	18887	Dev 4°
8	-19.6	20.9	-----	82.2	20.9	2.39	34.2	0.249	0.207	11360	
9	-21.1	30.3	-----	-30.6	-21.1	2.87	29.3	0.238	0.198	13922	
10	-12.6	18.3	-----	-27.4	-12.6	2.89	38.0	0.256	0.215	12290	

**Declaration**

This dissertation ‘Secure Indoor Navigation and Operation of Mobile Robotics’ is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work of this dissertation has been done by me under the guidance of Prof. Dr.-Ing. habil. Kerstin Thurow and Prof. Dr. -Ing. Norbert Stoll, at the University of Rostock, Germany. Also the dissertation has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Mazen Ghandour

Rostock, 01 December 2016

## Curriculum Vitae

### Personal Details

Name: Mazen Ghandour  
Date of birth: 10 May 1986  
Email address: mazen.ghandour@gmail.com  
Nationality: Syrian

### Education

2014- 2017                      Ph.D. student at the University of Rostock  
Research group “Mobile Robotics” under the supervisory of Prof.  
Dr.-Ing. habil. Kerstin Thurow and Prof. Dr. -Ing. Norbert Stoll.  
Research topic (Secure Indoor Navigation and Operation of  
Mobile Robotics)

2009-2012                      M.Sc. Mechatronics Engineering (Tishreen University-Syria)  
Master thesis was implemented under the supervisory of Prof.  
Dr. -Ing. Mohsen Dawood, topic (Development of Control  
and Surveillance Systems at 16<sup>th</sup> of November Dam)

2005-2009                      B.Sc. Mechatronics Engineering (Tishreen University-Syria)

### Work Experience

Jan 2013 – Oct 2013              Electrical Engineer (Dalmar Shipping LLC), Dubai – UAE  
- Developing electrical solutions for cargo ships.  
- Supervising the installation of electrical facilities.

Nov 2010 – Dec 2012              Mechatronics Engineer (Agob & Khatchadourian Automation  
Company), Aleppo – Syria.  
- PLC & SCADA Systems (Siemens).  
- Industrial Control and Automation Systems.

Jul 2010 - Dec 2012              Mechatronics Engineer (Tishreen University), Latakia – Syria  
- Teacher’s assistant.  
- Supervisor for several Mechatronics graduation projects.  
- Supervisor for Mechatronics & Robotics Labs.

Jul 2009-Jun 2010              CNC Engineer (Arabian Steel Company ASCO), Latakia-Syria  
- Programming the CNC machines (Lathe, Grinding, Notching).  
- Planning production and supervise the machines.

### Awards

- Deutscher Akademischer austauschdienst (DAAD) scholarship award for two years 2015-2017. Program (Research Grants – Doctoral Programmes in Germany).

### List of Publications

- **Ghandour, M.**, Liu, H., Stoll, N., Thurow, K. “*Improving the navigation of indoor mobile robots using Kalman filter,*” in Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International, 2015, pp. 1434–1439. (SCOPUS)
- **Ghandour, M.**, Liu, H., Stoll, N., Thurow, K. “*Interactive collision avoidance system for indoor mobile robots based on human-robot interaction,*” in 2016 9th International Conference on Human System Interactions (HSI), 2016, pp. 209–215. (SCOPUS)
- Liu, H., Stoll, N., Junginger, S., Zhang,J., **Ghandour, M.**, Thurow, K. “*Human-Mobile Robot Interaction in Laboratories Using Kinect Sensor and ELM Based Face Feature Recognition.*” In 2016 9th International Conference on Human System Interaction (HSI), 2016, pp. 179-202. (SCOPUS)
- **Ghandour, M.**, Liu, H., Stoll, N., Thurow, K. “*A Hybrid Collision Avoidance System for Indoor Mobile Robots based on Human-Robot Interaction,*” in 2016 17th International Conference on Mechatronics, Mechatronika 2016, (Accepted, in press, SCOPUS)